ENHANCED IPFIX FLOW PROCESSING MECHANISM FOR OVERLAY NETWORK MONITORING

SHAHZADA KHURRAM

DOCTOR OF PHILOSOPHY
UNIVERSITY UTARA MALAYSIA
2019

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

SHAHZADA KHURRAM

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"ENHANCED IPFIX FLOW PROCESSING MECHANISM FOR OVERLAY NETWORK MONITARING"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : ***25 April 2019.***
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:
April 25, 2019.

Pengerusi Viva: *(Chairman for VIVA)*	**Assoc. Prof. Dr. Yuhanis Yusof**	Tandatangan *(Signature)*
Pemeriksa Luar: *(External Examiner)*	**Prof. Dr. Haji Mazani Haji Manaf**	Tandatangan *(Signature)*
Pemeriksa Dalam: *(Internal Examiner)*	**Dr. Amran Ahmad**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Assoc. Prof. Dr. Osman Ghazali**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Dr. Shahrudin Awang Nor**	Tandatangan *(Signature)*

Tarikh:
(Date) ***April 25, 2019***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Pengkomputeran awan adalah teknologi yang baru muncul. Masyarakat menggunakan teknologi ini pada kadar yang lebih pantas, kerana trafik rangkaian awan ini berkembang pada kadar yang sukar untuk dikendalikan. Alat pemantauan adalah aspek penting dalam pengkomputeran awan dan menjadi lebih menyerlah dengan penggunaan perkhidmatan awan. Rangkaian tindanan menyediakan laluan baru untuk menumpu rangkaian dan bekerja sebagai rangkaian maya bebas di atas rangkaian fizikal. Pada masa kini, teknologi rangkaian tindanan awan dalam infrastruktur awan mempunyai jurang kebolehlihatan, yang bermaksud pembekal awan dan pengguna terlepas isu prestasi utama untuk mengatasi masalah trafik rangkaian tindanan. Justeru, untuk memastikan pengawasan rangkaian dan mengenalpasti potensi masalah, alat pemantauan rangkaian diperlukan untuk mengesan dan melaporkan lebih mendalam bukan sahaja untuk melihat trafik yang tersembunyi tetapi juga menyediakan maklumat berkaitan teknologi rangkaian tindanan awan yang khusus sesuai dengan pusat data skala besar moden. Oleh itu, kajian ini mencadangkan mekanisme peningkatan Eksport Maklumat Aliran IP (IPFIX), mengikuti Kaedah Penyelidikan Reka Bentuk untuk pengawasan rangkaian tindanan awan dengan mengadopsi teknik berasaskan aliran yang fleksibel. Tambahan pula, penyelesaian yang disediakan dalam penyelidikan ini terdiri daripada pelbagai mekanisme: mekanisme penapisan paket yang lebih baik melalui teknik penapisan perbandingan sifat dan teknik penapisan hash-based. Mekanisme klasifikasi aliran berasaskan Virtual Extensible Local Area Network (VXLAN), menggunakan bentuk aliran 6-tupel dan bentuk aliran yang diterima pakai. Mekanisma templat mesej IPFIX yang terdiri daripada kumpulan ruangan merekod data dalam sistem pemproses aliran IPFIX. Penemuan menunjukkan bahawa pendekatan yang dicadangkan dapat menganalisa trafik rangkaian tindanan *multi-tenant* untuk mengenal pasti, menjejaki, menganalisis dan terus memantau prestasi perkhidmatan rangkaian tindanan awan. Selain itu, mekanisme yang dicadangkan adalah sumber yang cekap di mana gabungan Mesej VFMFM+6tuple+VXLAN menggunakan 4.63% kurang CPU, manakala gabungan Mesej VHFM+AFCM+AFCM menggunakan 11.45% kurang CPU daripada IPFIX Standard. Sumbangan kajian ini akan membantu pengendali rangkaian awan dan pengguna akhir untuk menyelesaikan masalah prestasi berasaskan rangkaian tindanan dengan cepat dan secara proaktif dengan kebolehlihatan secara akhir-ke-akhir dan wawasan yang boleh dilakukan.

Kata kunci: Pengkomputeran awan, Rangkaian tindanan, Virtual Extensible Local Area Network, Pemantauan aliran paket.

Abstract

Cloud computing is an emerging technology. People are adopting cloud at a faster rate, due to this cloud network traffic is increasing at a pace which is challenging to manage. Monitoring tool is an essential aspect of cloud computing and becomes more apparent with the acquired of cloud services. Overlay network provides new path to converge network and run as an independent virtual network on top of physical network. Currently, cloud overlay network technologies in cloud infrastructure have visibility gaps, which mean cloud provider and consumers miss out the major performance issues for troubleshooting of overlay network traffic. Hence, to keep a close watch on network and catch potential problems, a network monitoring tool required, to track and report more in-depth for not only see the hidden traffic but also presents the related information of cloud overlay network technologies specifically suited to the modern cloud-scale data center. Therefore, this study proposes an enhanced IP Flow Information Export (IPFIX) mechanism for cloud overlay network monitoring by adopting flexible flow based technique. Furthermore, the solution provided in this research consist of diverse mechanisms: enhanced packet filtering mechanisms using property match filtering technique and hash-based filtering technique. Virtual Extensible Local Area Network (VXLAN) based flow classification mechanisms using 6-tuple flow pattern and adoptable flow patterns. IPFIX message template mechanisms, which is comprise set of fields for data records within the IPFIX flow processing system. The findings demonstrate that the proposed mechanism can capture multi-tenant overlay network traffic to identify, track, analyze and continuously monitor the performance of cloud overlay network services. The proposed mechanisms are resource efficient where the combination of VFMFM+6tuple+VXLAN Message consume 4.63% less CPU, while the combination of VHFM+AFCM+AFCM Message consume 11.45% less CPU than Standard IPFIX. The contributions of this study would help cloud network operators and end-users to quickly and proactively resolve any overlay network based on performance issues with end-to-end visibility and actionable insights.

Keywords: Cloud computing, Overlay networks, Virtual Extensible Local Area Network, Packet flow monitoring

Declaration

Some of the works presented in this thesis have been published or submitted as listed below.

[1] **S. Khurram**, O. Ghazali, F. Shahzad, A. S. Osman "A Survey of Cloud Monitoring: High Level, Low Level, Underlay and Overlay," *in 4th International Conference on Internet Applications Protocols and Services (NETAPPS2015)*, December 1-3, 2015, Cyberjaya, Malaysia.

[2] **S. Khurram** and O. Ghazali, "Design and Development of VXLAN Based Cloud Overlay Network Monitoring System and Environment", *Information Technology – New Generations. Advances in Intelligent Systems and Computing*, pp. 141-147, vol 738, Springer Nature America, 2018.

[3] **S. Khurram** and O. Ghazali, "A Comprehensive Survey of Cloud Monitoring", *European Journal of Computer Science and Information Technology(EJCSIT)*, pp. 51-65, vol 6, Issue 5, 2018.

[4] O. Ghazali and **S. Khurram**, "Enhanced IPFIX Flow Monitoring for VXLAN based Cloud Overlay Networks", *Conference on Mathematics, Informatics and Statistics (CMIS2018)*, October 29-31, 2018, Terengganu, Malaysia.

Acknowledgements

In the name of Allah the Most Beneficent, the Most Merciful.

The first gratitude I owe is profoundly to Almighty Allah (SWT) for giving me the strength and good health throughout my study period. Credit must go to my supervisors Prof. Madya Dr. Osman Ghazali and Dr. Shahrudin bin Awang Nor whose instructive guidance, encouragement and relentless support enabled me to complete successfully this study. From conceptualization to conclusion, you have been amazing in supervising this work. I am heartily grateful. Indeed, I look forward to working with you in the nearest future. I render to you a special and sincere debt. You were a mentor because you were more than a supervisor to me. You taught me the true meaning of humility and kindness. God bless you Prof. Madya Dr. Osman Ghazali. I shall, and forever remain grateful to you! I am also greatly indebted to my external examiners Prof. Dr. Haji Mazani Haji Manaf and internal examiner Dr. Amran Ahmad for their constructive criticism and instructive guidance.

I also wish to acknowledge the research informants who participated in this study for their commitment. Many thanks to Emily Sarneso from Carnegie Mellon University help me for development of Vxlan based Plugin in YAF. Many thanks also go to my mentor of several years, Dr. Mujahid Alam. Thank you so much for your scholarly support, I appreciate you so much. Thank you so much for your love and kindness.

Finally, I wish to express special thanks to my colleagues and friends Dr Tanveer Husain, Dr Dost Muhammad, Faisal Shahzad, Tareef Ali Khan and Ali Naeem. You gave me love, support and strength, may Allah always bless all of you. (Ameen).

Dedication

This dissertation is nicely dedicated to my father the late Rana Shaukat Ali Khan, my mother the late. Shahida Perveen, may Allah reward you with Jannah!

To my beloved wife Saiqa Sadiq and my kids Muhammad Ahyan Khurram and Hibba Khurram your love, patience, words of encouragement and prayers were the best tonic that continued to soothe the fatigue that was always felt.

Finally, it is to Allah who gave me life and strength to undertake this study that most importantly deserves the highest praise and honors.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
IaaS	Infrastructure as a Service
IP	Internet Protocol
IPFIX	IP Flow Information Export
LAN	Local Area Network
MAC	Media Access Control
MIB	Management Information Database
NMS	Network Management System
NVGRE	Network Virtualization with Generic Routing Encapsulation
PaaS	Platform as a Service
QoS	Quality of Service
SaaS	Software as a Service
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
STT	Stateless Transport Tunneling Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNI	Virtual Network Identifier
VTEP	Virtual Terminal End Point
VXLAN	Virtual eXtensible Local Area Network
WAN	Wide Area Network
VHFM	VXLAN based Hash Filtering Mechanism
VFMFM	VXLAN Field Match Filtering Mechanism
AFCM	Adaptable Flow Classification Mechanism

CHAPTER ONE
OVERVIEW

This chapter presents a brief introduction to the proposed research. This chapter also presents the general background information of cloud computing along with cloud monitoring and brief overview of cloud overlay networks. The chapter also outlines the problem statement and research questions, research motivation, research objectives, research scope and the significance of the research along with the expected contribution. Finally, the outline of the proposal is presented at the end.

1.1 Background

Cloud computing provide the various computing resources as a service. It is the current iteration of utility computing and returns to the model of resource sharing. The terms “cloud computing” and “cloud” have previously been contentious. According to National Institute of Standards and Technology (NIST)’s definition: “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1]. Cloud terminology has largely become standardized and has entered the academic lexicon. Today, cloud computing underpins a significant

portion of the web and is the de facto means of deploying services and applications at scale.

The cloud can be further divided into four different implementation models: Public, Private, Hybrid and Community based on whether it's created internally, outsourced or a combination of the two [1]. Cloud computing is made up of three layers Software-as-a-Service (SaaS) users are provided access to application software run on distant virtual machine (VM) via internet, Platform-as-a-Service (PaaS) cloud providers deliver a computing platform, typically including operating system for execution of required applications and Infrastructure-as-a-Service (IaaS) provide computing resources including servers, networking, storage, and data center space on a pay-per-use basis [1]. To acquire cloud layers services, a Service Level Agreement (SLA) between the cloud provider and the customer is required. The acquired services based on the SLA should be monitored continuously.

Monitoring is the process of observing and tracking applications as well as resources at run time. Fatema et al. [2] provides a brief definition of the term monitoring as *"a process that fully and precisely identifies the root cause of an event by capturing the correct information at the right time and at the lowest cost in order to determine the state of a system and to surface the status in a timely and meaningful manner"*. A multi-tenant nature of cloud can be challenging for smooth management in term of performance constraints and quality of service because the services of the cloud are scalable, flexible and on demand. Monitoring plays an important role in the utilization of cloud resources at all layers. The main features for efficient monitoring in cloud environment are: Capacity and resource planning,

Capacity and resource management, Data center management, SLA management, Billing, Troubleshooting, Performance management and Security management [3].

Due to the dynamic nature of resources provisioning and allocation in cloud environment. The performance of the network utilization is unpredictable [4]. Although progress has been made for cloud monitoring over the past years [5]–[7], however it still faces some challenges to monitoring in cloud environment efficiently. The current monitoring solutions, including many of those in the open-source domain are relied on low level monitoring, high level monitoring and underlay network monitoring concepts. [8]

1.2 Cloud Overlay Network

Traditional cloud providers are struggling to keep up with new computing requirements for example VM migration, scalability and network isolation in large cloud network environment [9]. Network architects should rethink their designs and adopt simpler topologies and new control protocols to achieve better performance and operational agility in multi-tenant cloud networks. The aim of an overlay network is the decoupling of the physical topology from the logical topology, to allow connectivity between compute (virtual or physical) and network (virtual or physical) regardless of where these may reside within the cloud data center. This approach delivers an optimal level of flexibility and mobility, so that computing nodes can now be dynamically placed anywhere in the cloud, removing the traditional layer-2 boundaries of the physical infrastructure. Below are the problems in traditional cloud networks [10].

i. VM Migration: In cloud computing environment VMs can move from one host to another for various reasons. For instance, distribute the workloads, CPU overload, insufficient memory or host failure. To ensure uninterrupted services during VM migration, the IP and MAC addresses of VMs must remain unchanged. To meet this requirement, the service network must be a Layer 2 network that provides multipath redundancy and reliability.

ii. Scalability: Layer-2 networks are built using Ethernet switches and are not scalable because these use flooding-based source media access control (MAC) learning which does not scale beyond a couple of hundred hosts. On a large cloud Layer-2 network, data packets are forwarded based on MAC address entries. Therefore, the number of VMs supported on the network depends on the MAC address table size.

iii. Network Isolation: Network isolation capabilities are limited. Most networks use virtual local area network (VLANs) or virtual private networks (VPNs) for network isolation [11]. However, these two network isolation technologies have the following limitations on largescale virtualized networks:

a) The VLAN tag field, as defined in IEEE 802.1Q, has only 12 bits, and can only identify a maximum of 4094 VLANs, making it insufficient for identifying users on large Layer 2 cloud networks.
b) VLANs or VPNs cannot support dynamic network adjustment on traditional Layer 2 networks.

Traditional approaches of building layer-2 Ethernet switch networks or connecting islands of Ethernet LANs using layer-3 switches, fail to achieve their cloud network goals for example VM migration, scalability and network isolation in large cloud network environment. To overcome this problem, overlay tunneling techniques introduces in cloud environment which can encapsulate layer 2 frames into layer 3 IP packets. Overlay network protocols address the above problems on large Layer-2 cloud networks as follows:

i. When overlay network protocols are used to construct a large Layer-2 cloud network, Virtual machine IP and MAC addresses can remain unchanged after VM migration.

ii. Overlay network protocols encapsulate data packets sent from VMs into UDP packets and encapsulate IP and MAC addresses used on the physical network into outer headers. The network is only aware of the encapsulated parameters. This greatly reduces the number of MAC address entries required on large Layer 2 cloud networks.

iii. Overlay network technology VXLAN uses a network identifier (VNI) field similar to the VLAN ID field defined in IEEE 802.1Q. The VNI field has 24 bits and can identify a maximum of 16M VXLAN segments [12]

Few standards have been proposed to enable overlay networks: Virtual eXtensible LAN (VXLAN) [13], Network Virtualization with GRE (NVGRE) [14] and

Stateless Transport Tunneling Protocol (STT) [15]. These overlay protocols use different encapsulation techniques to solve current network limitations. Figure 1.1 presents a possible cloud overlay network method for communication in large cloud networks.

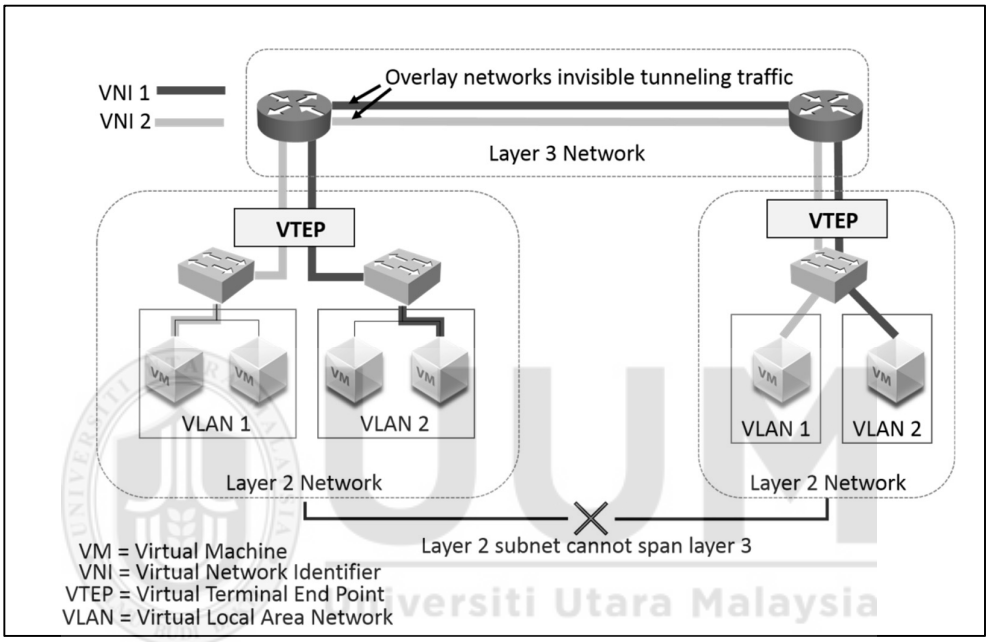

Figure 1.1. Cloud Overlay Network Method for Communication in Large Cloud Environment [12].

1.3 Motivation

Cloud services become more popular because of extremely cost effective, scalability, flexibility, network and storage capacity, increased data reliability, agility, tension free maintenance and management. These cloud services are rapidly growing day by day due to fast adoption and migration of workload from private data centers to cloud data centers. In addition, cloud data centers should handle higher traffic load due to this workload migration [7]. In cloud environment service

providers use virtualization and automation for cloud services. Therefore, cloud data centers required efficient cloud network traffic handling for higher capacity, increased performance and great throughput. Cloud network plays important role to migrating and storing the workload from private data center to cloud data centers. According to the Cisco report annual global cloud IP traffic will reach 8.6 ZB globally by 2019 [16]. Cloud traffic is expected 33% grow by annual growth rate (Compound Annual Growth Rate, CAGR) and by 2019 86% of workloads will be processed by cloud data center while only 14% will be processed by traditional data centers with 8% CAGR. Figure 1.2 shows global annual cloud traffic growth statistics till 2019 [16].

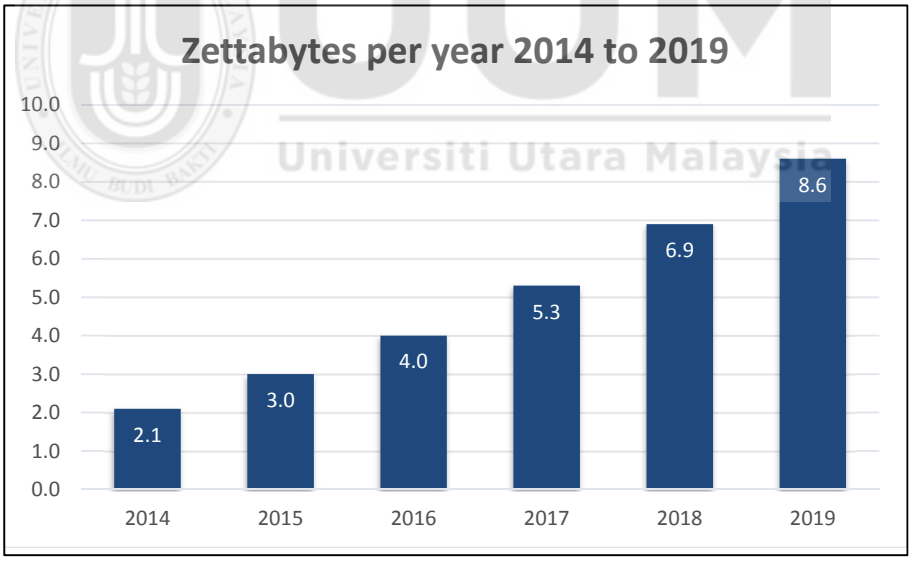

Figure 1.2. Global Annual Cloud Traffic Growth [16].

Network traffic growing rapidly in cloud data centers, from technical point of view the extent of this growth is not discussed. Due to dynamic resource provisioning

and high speed virtualized cloud networks, growing of cloud network traffic also creating problem to manage it efficiently by service provider and end user prospectus. Therefore, cloud network performance monitoring system is required to monitor, troubleshoot, and analyze what is happening across your enterprise network environment. Hence, it is highly recommended to quickly and proactively resolve any network-based performance issues with end-to-end visibility and actionable insights.

1.4 Problem Statement

Monitoring is an essential aspect of any scientific field especially for cloud computing and Internet of Things [17] where a large number of users or customers are involved. So, requirement of monitoring tools become more obvious with the acquired of cloud computing services. More and more individuals and organizations are adopting cloud at a faster rate, due to which cloud traffic is increasing at a pace which is difficult to manage [7]. To manage large and complex cloud network infrastructure, the monitoring system should be able to precisely capture its state [18]. The virtualization plays a vital role to implement cloud computing, but virtualization technologies add an additional level of complexity for the consumers and cloud providers. It leads to manage physical and as well as virtual resources of cloud infrastructure [9, 15–17].

The complex cloud network infrastructure requires root cause analysis of network problems and troubleshooting in depth. To find the cause of problem may be searched several layers including physical and virtual layers that may be time

consuming. Therefore, a reliable and a real time monitoring system is required for both cloud provider and consumer to understand the performance issues and failure causes in cloud infrastructure [22]. Likewise, an organization may have mission critical application and want to host application on multiple clouds for high availability and work load sharing concerns. Thus, monitoring is essential to significantly improve the performance of real time applications and troubleshooting for multiple cloud network infrastructure [23].

There have been many research and development efforts in the field of cloud monitoring and traffic analysis for last few years. As a result, many tools were introduced to meet various objectives of cloud traffic measurement. These tools have abilities to monitor high level, low level and underlay networks but none of them meet the requirement of cloud overlay technologies monitoring. It may be because of cloud overlay technologies are still new and emerging stage.

The monitoring systems for resources utilization in virtualized and large cloud environment recently proposed [5, 6] . However, these mechanisms do not address the complete picture of monitoring in respect of cloud overlay networks in virtualized environment. Moreover, these mechanisms have not taken the dynamic nature of cloud overlay network performance into account. The SNMP based cloud network monitoring systems proposed [7, 20, 21]. However, these monitoring systems not fulfil the monitoring requirement of high speed and dynamic cloud network environment due to SNMP has many known limitation to collection of monitoring data by polling technique and delayed response caused by jitter [22, 23]. For classifying traffic into flows, the real time cloud monitoring architecture

based on network probes proposed by L. Deri and F. Fusco [28]. They do not include the mechanism of overlay network traffic classification in proposed architecture. Mann et al. [29] proposed flow-based network service monitoring solution for cloud infrastructure. Author only analyzed flow monitoring protocols such as NetFlow [30] and sFlow [31] on physical switches and virtual switches for traffic analysis. However, author has not taken the dynamic nature of cloud overlay network performance into account.

Cloud overlay technology introduces the same visibility challenges as most exist for encapsulation methods. Essentially, end-to-end traffic is hidden inside the tunnel, so it must be able to strip away the encapsulation for sustained monitoring and troubleshooting. Currently overlay network technology in cloud infrastructure have visibility gaps, which mean cloud provider and consumers can miss out the major performance issues for troubleshooting of overlay network traffic. Thus, to keep a close watch on network and catch potential problems, an urgent need of network monitoring tool to dynamically track and report more in-depth for not only see the invisible traffic but also presents the related information of cloud overlay network technologies.

In the scope of cloud overlay network monitoring, there is need to export captured per flow data for further analysis and measurement. The Internet Engineering Task Force (IETF) introduced IP Flow Information Export (IPFIX) protocol for exporting per flow information. Traditional IPFIX mechanism not able to capture traffic and monitor cloud overlay networks. As a result the IPFIX architecture described in [28, 29, 30] needs to be enhanced. This enhanced process needs to

fulfill some requirements from the data manipulation point of view. The enhanced process should provide various functions like aggregation, filtering, or the modification of flow records for the means of saving system resources and providing processing tasks for the collecting of only cloud overlay network traffic data. To improve productivity and efficiency in large cloud computing network environment, there is a need of enhance the traditional IPFIX flow processing system for cloud overlay network monitoring.

1.5 Research Questions

In order to design the cloud overlay network monitoring system and analysis of cloud overlay traffic, the work has been organized to find answers to the following questions.

Question 1: How to dynamically track and monitor the overlay traffic in cloud network infrastructure?

Question 2: How to enhance current IPFIX flow processing mechanism for overlay network monitoring in large and complex cloud network environment?

Question 3: How to evaluate the performance of network traffic using proposed mechanism for cloud overlay network monitoring?

1.6 Research Objectives

The aim of this research is to come up with mechanisms to enhanced current IPFIX flow processing that can help to monitor, track and analyze the multi-tenant traffic in large cloud overlay network environment. The proposed mechanisms must be able to identify and filter the overlay packets in large and complex cloud network environment. In order to enhance the current flow aggregation mechanism in flow process system it should be support overlay packet pattern for flow classification within IPFIX flow processing environment. Furthermore, the proposed mechanism should be able to support the overlay packet templates for data records IPFIX messages within the IPFIX flow processing system. This aim could be further explained with the aid of the following specific research objects. :

Objective 1: To design the technique that strip away encapsulation, identify and filter the cloud overlay networks packets in high speed cloud network environment.

Objective 2: To enhance the current IPFIX based flow processing mechanism that can real time monitor, track and analyze the cloud overlay network traffic in large cloud network infrastructure.

Objective 3: To evaluate the multi-tenant network traffic performance in large cloud network infrastructure.

1.7 Research Scope

The overall goal of this research is to develop mechanisms for cloud overlay network monitoring. It includes designing, developing, implementation and testing of monitoring for cloud overlay networks performance in high speed and large cloud network environment. The proposed monitoring system can real time track and monitor the cloud overlay network traffic and also can be used in depth performance analysis and troubleshooting of cloud overlay networks issues in large cloud network infrastructure.

The scope of this research to enhanced traditional IPFIX mechanisms for cloud overlay network monitoring. In addition, it is envisaged to come up with mechanisms, techniques and monitoring overlay traffic for meeting the objectives given in section 1.6 . Finally, the proposed design may not monitor all cloud overlay network technologies. As cloud overlay network technologies are still new and emerging, thus, only mature and widely adopted technology in cloud data centers [35] Virtual eXtensible LANs (VXLAN) is considered for cloud overlay network monitoring.

1.8 Significance of the Research and Expected Contributions

Cloud overlay is still a new and emerging technology and is being adopted in cloud network infrastructure, especially for virtual networking in the hypervisor for virtual machine to virtual machine communication. In this research we propose a design of cloud overlay network monitoring with enhanced IPFIX flow processing mechanism that can continuously monitor cloud overlay networks traffic. The

significance of this work is that, once the objective mentions in Section 1.6 have been achieved. It would help multiple stakeholders related to the cloud computing including consumers, enterprises, service providers and cloud brokers. The proposed monitoring system can real time track and monitor the cloud overlay network traffic and also can be used in depth performance analysis and troubleshooting of cloud overlay networks issues in large cloud network infrastructure. The mechanism developed as part of this research will be extensible from single cloud environment to multi cloud environment. The contributions of this research are summarized as follows:

- A monitoring system that can real time track and monitor the cloud overlay network traffic in large cloud network infrastructure.
- Provide a technique that striped out encapsulation, identify the cloud overlay networks packets in high speed cloud network environment.
- Proposed mechanism that can be used in depth performance analysis and troubleshooting of cloud overlay multi-tenant networks issues.

1.9 Organization of the Thesis

This thesis has been organized into six chapters. Chapter one provides an overview of the overall research works including the outlines of the research motivation, problem statement and research questions, research objectives, scope of the research and the significance of the research along with the contributions.

Chapter Two critically evaluates and summarizes the literature which is relevant to the topic of the study. The chapter provides the background information on cloud computing along with discussion on recent published work. The chapter also includes an in depth discussion on the cloud overlay network technologies along with virtualization implementation in cloud environment. More emphasis is given to cloud monitoring and its types with detail discussion. This Chapter also explained in detail related works and popular architectures of cloud network monitoring. Finally, an in depth analysis is presented on the cloud monitoring systems proposed for cloud environment.

Chapter Three establishes the research methodology adopted in this research work. The design research methodology has been adopted to suit the requirements of this work. The details of every step along with approaches used within those steps have been explained in detail.

Chapter Four present the performance of flow technologies within real time VXLAN based cloud overlay network environment. Since VXLAN is new technology in cloud overlay network environment therefore, no simulation tool available to paradigm VXLAN based cloud overlay network environment. Hence, in this chapter discusses how cloud overlay network environment can be modeled for monitoring. Finally, using benchmark methodology performance analysis of flow technologies evaluated and standardize the IPFIX performance results in order to comparing the results with our proposed monitoring mechanism for evaluation and validation purpose.

Chapter Five explains the enhanced packet capturing and filtering mechanism for cloud overlay networks. Two mechanisms VXLAN Field Match Filtering Mechanism (VFMFM) and VXLAN based Hash Filtering Mechanism (VHFM) have been developed that strip away encapsulation, identify and filter the cloud overlay networks packets in high speed cloud network environment. The complete process from packet observation to selection has been discussed in detail. The proposed mechanisms were tested on hybrid simulation environment and collected results compare with standard IPFIX monitoring system.

Chapter Six explains the enhanced IPFIX flow processing mechanism for cloud overlay networks. All the mechanisms from 6-tuple based flow classification to adoptable flow classification and VXLAN based messages to adoptable flow classification mechanism messages for IPFIX data records have been developed that contributed to the final contributions were explain in detail in this chapter. This chapter also present the performance evaluation of proposed monitoring system and compare with standard monitoring method that has been used for validating the proposed work.

Chapter Seven concludes this thesis by summarizing the research along with contribution, and providing an outline of the limitations of this work, and some suggestions for future work based on findings of the study.

CHAPTER TWO
LITERATURE REVIEW

This chapter explains the cloud monitoring systems and its types. The literature review enables to understand cloud monitoring and its techniques in detail specifically for cloud overlay networks. Section 2.2 portrays cloud computing in detail with cloud services models, cloud deployment models and cloud virtualization, Section 2.3 presents software defined networking, Section 2.4 describes the cloud overlay networks in detail, Section 2.5 explains cloud monitoring and types of cloud monitoring with detail discussion of current cloud monitoring systems while Section 2.6 explains detailed network monitoring techniques including flow based monitoring technology.

2.1 Cloud Computing

Cloud computing, where large collection of remote servers is networked, facilitates on demand computing resources available to everyone over the Internet. On demand cloud services available to end users and organizations as per requirement, including application, storage and servers. Consumers can access these services through internet via web browser or APIs. Therefore, the applications, storage, servers and infrastructure does not reside at consumer and organization end. As a result, without any capital investment and worries of installation or maintenance of servers and networking, end users only focus on the resource utilization for its purpose which acquired from a cloud provider as per requirement. Cloud consumers who utilize these resources would be required to pay only for services they accessed. Cloud

services are easy to manage and faster to deployment for individuals and Small and medium business organization who can start online business without worries of hardware and infrastructure installation with low upfront cost. Cloud computing have a lot of benefits including scalability, where users can add and remove computing resources on demand. Consumers can also get advantage from cloud services in the shape of improved efficiency and high availability according to their requirement. Workload and unpredictable growth of business often create problem to extend the IT services thus, organizations can get benefit from cloud services because the cloud share their large pool of computing resources as a self-services.

2.1.1 Cloud Services Models

Cloud computing is made up of three delivering models currently in the cloud market. These are Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) [1].

2.1.1.1 Software as a service (SaaS)

In SaaS model software application access provided on lease basis according to the user requirement. Consumer can access these applications using internet via web browser or any internet enable device which can be single application or many applications as per required. Users can select application which they want to install as many cloud providers provide pre-defined application in the case of public cloud on the other hand in private cloud user can install application own their premises. Individual user and particularly small and medium business users which they have limited resources and budget constraints, SaaS is an affordable choice for business

start-up. In SaaS, there is no need for maintenance of software application, software licensing, managing of cloud platform and infrastructure. Facebook, google apps, twitter, instant messaging and webmail is the example of SaaS.

2.1.1.2 Platform as a service (PaaS)

In PaaS model application development environment provided to the developers. Cloud provider provide computing platform including operation system and other software tools with execution environment according to the user requirement typically database and web server. Application developers can build and execute the application for their need. Before that organizations required IT team to deploy the business tool for their requirement own their premises. They also need to install hardware, server operating software including web server, databases and the actual application to development of new software and testing. Hence, their IT team maintain all of these resource over the time. Under the PaaS model all these resources maintained by PaaS provider user only need to access through internet via web browser to login and start platform. User can upgrade and downgrade resources according to the application development requirement. Cloud provider charge usually pay per usage for resources utilization of PaaS services. App server, database server and web servers are example of PaaS.

2.1.1.3 Infrastructure as a service (IaaS)

IaaS the basic cloud service model which provide on demand compute services including physical and virtual machines (VMs), memory, storage devices, network devices, datacenter space and infrastructure to the end users. To enabling virtualized

services using physical resources a hypervisor or virtual manager required like KVM, Xen, Hyper-V and VMware ESX etc. using the clustering technique with the hypervisor multiple physical machines resources are combined and these resources are allocated to the pool of VMs. Other services of IaaS are IP addresses, firewalls, virtual local area networks (VLANs) and load balancers, physical switches of layer 2 and 3 are using for network virtualization. Cloud provider offer these resources on demand form their large pool of equipment installed at datacenter. But the consumer are responsible for all these service maintenance including software and monitoring of their on demand infrastructure. The main advantage of IaaS is its highly scalable and faster to deployment according to the user requirement no need to wait for expensive hardware procurement process. Cloud providers charge on hour basis as per resources utilizations. Amazon Web Services (AWS), Google Compute Engine, Windows Azure and Rackspace are leading IaaS cloud service providers.

2.1.2 Cloud Deployment Models

Cloud deployment models are categorized on the basis of the location whether it's created internally, outsourced or a combination of the two. Cloud can be divided into four different implementation models Public, Private, Hybrid and Community based [1]. Individuals and different organization may select these of deployment models according to the requirements for instance security concerns, efficiency, scalability, budget constraints and management policy.

2.1.2.1 Virtualization

The virtualization is the enabling technology that makes cloud computing possible. Cloud uses hardware as well as software virtualization, storage and network virtualization. Hardware virtualization is the technology that either combines or divides computing, storage and network resources in order to provide an environment different from the real physical one [36] . Virtualization may combine multiple independent computer servers and physical network devices to present it as a single large system or divide a single computer into multiple functional environments where each one can behave like a single complete computing unit. Virtualization is achieved through techniques such as hardware, software partitioning or time sharing between multiple functioning logical systems.

In cloud computing environment, multi-tenant virtualization is used where a single system is presented as multiple virtual systems that can spawned and removed at will on demand. The virtualized infrastructure is created by installing a Virtual Machine Manager (VMM) on the physical hardware. The VMM provides the necessary isolation and security between the multiple virtual machines running in parallel on a single physical server. Cloud hosted physical servers can be allocated to many customers. When a single server is allocated to multiple customers, and many virtual machines are hosted on a single server simultaneously, its performance starts degrading due to the competition for resources between the hosted systems. Therefore, in order to maintain the service quality, the maximum number of virtual machines hosted on a system must be limited. The cloud virtualization divided into three main categories server virtualization, storage virtualization and network

virtualization. Furthermore, both virtual storage and virtual servers are connected through virtual network in cloud orchestration as shown in Figure 2.1. Therefore, network virtualization plays very significant role in cloud environment.

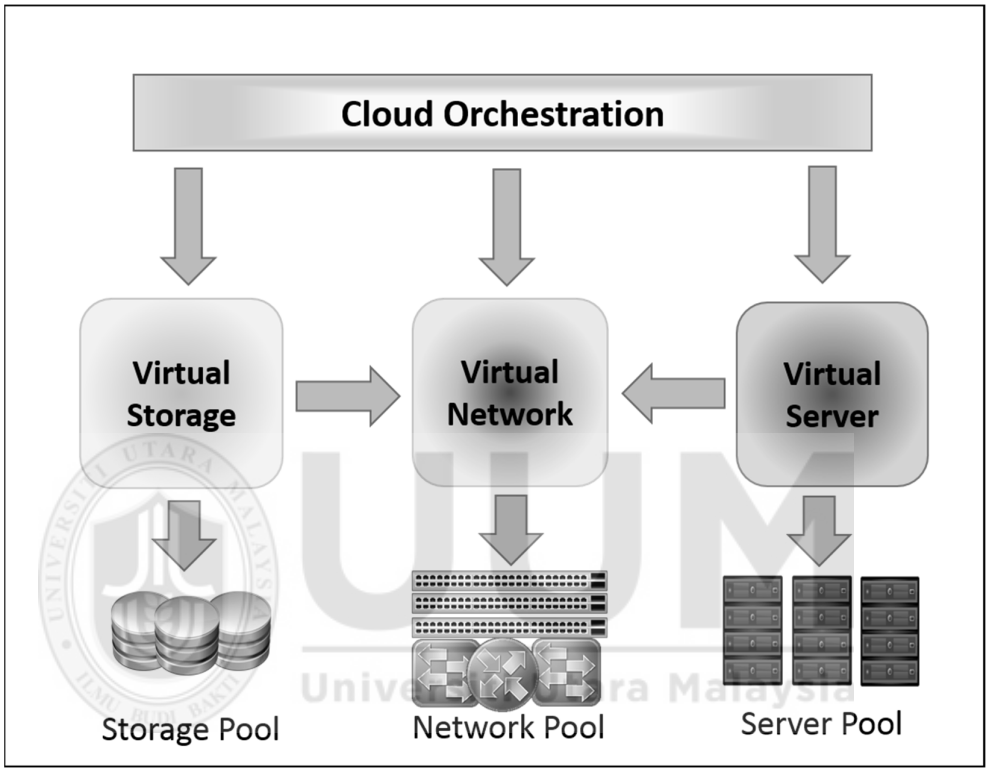

Figure 2.1. Cloud Orchestration [36].

2.1.2.2 Network Virtualization

Network virtualization refers to the combination of available physical network resources using hypervisor software into a single virtual network [37]. Available bandwidth divided into different virtual secure channels and that may assigned to any virtual machine in a cloud computing environment. Network virtualization allows running of isolated logical networks on a shared physical network. It consists of a combination of multiple network resources, capabilities and functionalities into

a single unit known as a virtual network. It is the solution for expanding data center devices that connect each other within virtualized environment. When network virtualization has been implemented, it hides the details of the network implementation and provides a unified view that can be customized through a software interface to according to the user requirements. Virtual private network (VPN) and virtual local area network (VLAN) are examples of network virtualization.

The advantages of network virtualization include:

- Infrastructure Utilization
- Infrastructure is shared between many different users or purposes
- Reduces infrastructure and energy cost
- Scalability
- Easy to extend resources in need
- Administrator can dynamically create or delete virtual network resources
- Agility
- Enables automation of network services establishment
- Network services can be orchestrated together with other IT infrastructure
- Resilience
- Virtual network will automatically redirect packets by redundant links
- In case of disaster, the virtual network can be easily recreated on new physical infrastructure
- Security
- Increased data traffic isolation and user segmentation

- Virtual network should work with firewall software

2.2 Software Defined Networking (SDN)

Software defined networking refers to the dynamic control of network devices by software programming to improve the network efficiency, flexibility and scalability. Software defined networking architecture is based on centralized controller, and network devices which have control plane and data plane. To manage the network devices and control functions, such as route calculation of network devices, are centralized on one controller, which generates the forwarding table and delivers it to network devices while each network device control plane is separated from the data plane. Network devices such as routers and switches are responsible only for forwarding packets and follow policies as defined in centralized controller. Software defined networking widely used in cloud infrastructure to enable centralized management of cloud tenant network control [38].

2.3 Cloud Overlay Network

Cloud data centers growth also rapidly increase the number of virtual machines deployment. VMs can move from one host to another for various reasons, for instance, distribution of workloads or host failure. These moves required basic configuration of VLAN trunking in cloud networked switches. IP addressing and VLAN are often assigned to virtual machines. They are being limited by the broadcasts domains in Layer 2, though VLAN can only identify a maximum of 4094 VLANs. However, advance and large cloud networks have thousands of switches that interconnect thousands of VMs that making it insufficient for identifying users

on large Layer-2 cloud networks. To overcome this problem, overlay tunneling techniques introduced in a cloud environment, which can encapsulate Layer-2 frames into Layer-3 IP packets.

Overlay network allow cloud providers and end users to orchestrate networks along with other virtual resources in cloud environment. It provides new path to converged network and run as independently virtual network on top of physical network. The cloud overlay network allows network resources to be dynamically provisioned similarly to virtual storage and virtual compute. Cloud overlay network technology is used in cloud data centers, to effectively isolate multiple tenants and automate network-wide virtual machine migration that fully satisfy the requirements of large cloud service providers and enterprises. A few standards have been proposed to enable cloud overlay networks, which include Virtual extensible LANs (VXLAN) [13], Network Virtualization with GRE (NVGRE) [14] and Stateless Transport Tunneling Protocol (STT) [15]. These cloud overlay protocols use different encapsulation techniques to overcome the current network limitations. The goal of cloud overlay network monitoring is to improve efficiency in large cloud computing environment. These overlay network protocols discussed in detail bellow.

2.3.1 Virtual eXtensible LANs (VXLAN)

VXLAN is a new overlay network protocol that uses tunneling technology for MAC-in-UDP encapsulation to extend large Layer-2 networks onto Layer-3 networks and defined in RFC 7348 [13]. Each VXLAN is identified with a 24-bit VNI. VXLAN encapsulation enables the layer-2 to communicate with any end point as long as the

end points are in the same VXLAN segment. These end points may not necessarily be in the same IP subnet, so the problem of 4094 VLANs and limited MAC address capacity in switches are eliminated.

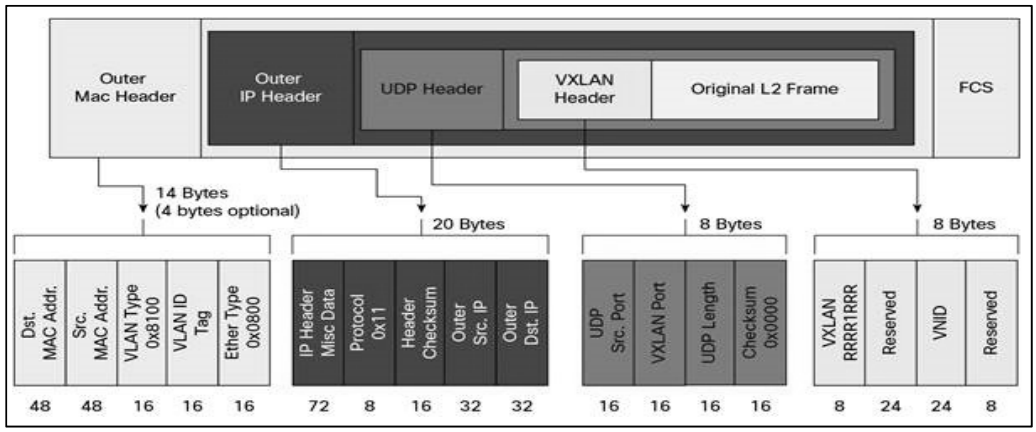

Figure 2.2. VXLAN Frame Format [39].

An 8-byte VXLAN header that consists of a 24-bit Virtual Network Identifier (VNID) and some reserved bits as shown in Figure 2.1. The VXLAN header and original Ethernet frame together with in the UDP payload. The 24-bit VNID is used to identify Layer 2 segments and to maintain Layer 2 isolation between the segments.

The benefits of VXLAN include:

When server virtualization is widely deployed in data centers' based on physical network infrastructure, VXLAN offers the following benefits:

i. Supports a maximum of 16M VXLAN segments with 24-bit VNIs, so a data center can accommodate a large number of tenants.

ii. Reduces the number of MAC addresses that network devices need to learn and enhances network performance because only devices at the edge of the VXLAN network need to identify VM MAC addresses.

iii. Extends Layer 2 networks using MAC-in-UDP encapsulation and decouples physical and virtual networks.

iv. Tenants can plan their own virtual networks, without being limited by the physical network IP addresses or broadcast domains. This greatly simplifies network management.

2.3.2 Network Virtualization Using Generic Routing Encapsulation (NVGRE)

NVGRE uses the GRE tunneling protocol encapsulation, defined in RFC 7637 [14], to create layer-2 network onto a layer-3 network. In NVGRE, address learning is implemented by the control plane. Compared with VXLAN, NVGRE is defective in terms of load sharing, i.e., NVGRE cannot implement GRE key-based load sharing. In addition, NVGRE tunnels are end-to-end, so the number of tunnels increases exponentially as the number of terminals increases. As a result, the overhead for tunnel maintenance becomes very large. Figure 2.3. describe the NVGRE Encapsulation Frame Format.

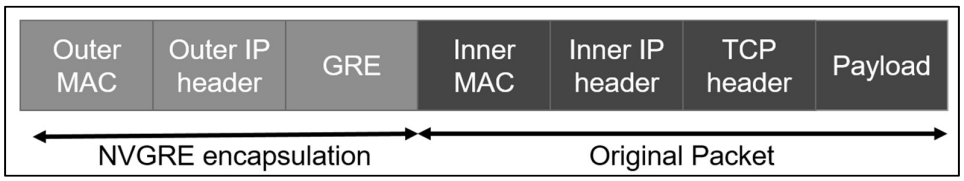

Figure 2.3. NVGRE Encapsulation Frame Format [40].

2.3.3 Stateless Transport Tunneling (STT)

STT is also an overlay technology used to create a layer-2 virtual network over a layer-2 or layer-3 physical network [15]. In technical terms, STT is very similar to VXLAN like VNI of STT is also 24-bit. STT has a multipath advantage by controlling transmission source packet headers. The difference between STT and VXLAN is that STT fragments data frames before encapsulation. Thus, the hardware acceleration of network cards can be fully utilized for higher efficiency. In addition, STT disguises STT packets as TCP/IP packets, and TCP packet headers do not maintain TCP state information; thus, retransmission does not occur after packet loss. In this way, STT tunnels are less reliable. Until now STT has not define as IETF standard protocol. Figure 2.4 describe the STT Encapsulation Frame Format

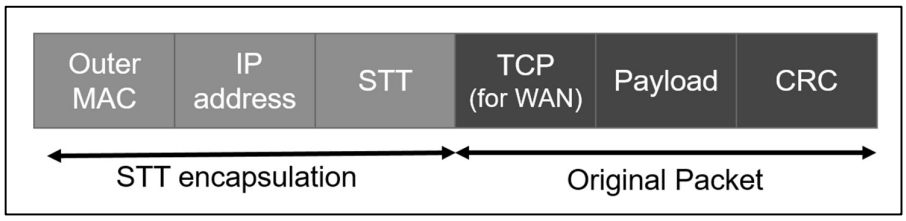

Figure 2.4. STT Encapsulation Frame Format [41].

2.4 Cloud Monitoring

Cloud monitoring is an undertaking grade solution that helps to maintain applications active and performing well constantly. A multi-tenant nature of cloud can be challenging for smooth management in terms of performance constraints and quality of service because the services of cloud are scalable, flexible and on demand. Monitoring can play important role in utilization of cloud resources of all layers. Cloud monitoring is essential for both cloud users and provider. Along with, it is a central tool for managing software and hardware infrastructure. Moreover, it furnishes information and key performance indicators for cloud layer services. In cloud computing, monitoring has two types namely high level and low level monitoring. Whereas high level monitoring is relevant to virtual platform information while low level is related to the physical infrastructure. Furthermore, two types of cloud network monitoring are for underlay and overlay networks.

2.4.1 Types of Cloud Monitoring

In this section, we discuss types of cloud monitoring. Cloud monitoring is divided into four categories: high level monitoring, low level monitoring, underlay network monitoring, and overlay network monitoring. Figure 2.5 presents the types of monitoring with relevant cloud layers.

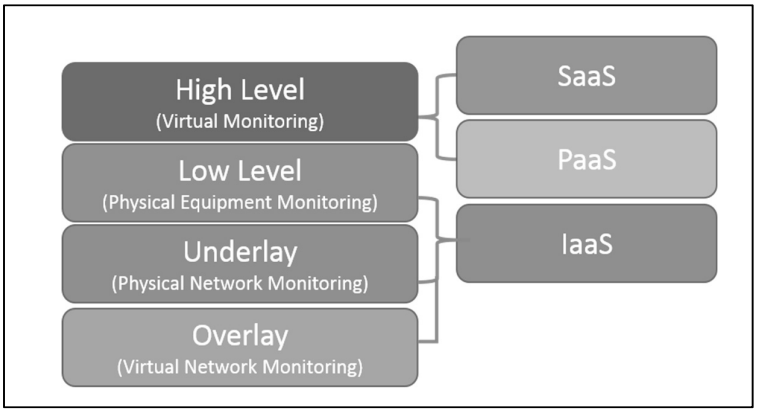

Figure 2.5. Types of Cloud Monitoring with Cloud Layers [3].

2.4.1.1 High Level

High-level monitoring collects data that is relevant to virtualized platform in cloud computing environment [3]. In cloud layers, monitoring SaaS and PaaS are considered as high-level monitoring. In SaaS environment, users can access cloud applications through the Internet. Cloud host applications can perform an extensive range of tasks for consumers. Facebook, webmail and instant messaging are examples of SaaS. Consumers are able to access the services through any internet enabled device. On the other hand, developers can build applications according to their requirements using PaaS environment. Web servers, App servers and Database management are examples of PaaS.

2.4.1.2 Low Level

Low-level monitoring collects data that is relevant to the physical infrastructure of the cloud [3]. In cloud monitoring, IaaS layer is considered as low-level monitoring. IaaS provides data center resources containing storage, servers, and datacenter space on a pay per use basis. Low-level monitoring is divided into two categories namely computation based and network based [42]. A collection of computation based

matrices is called low-level monitoring. Meanwhile, network-based monitoring in IaaS is divided into underlay network monitoring and overlay network monitoring.

2.4.1.3 Underlay

Collection of monitoring metrics relevant to the physical network in cloud computing environment called underlay network monitoring. Underlay monitoring is also a part of IaaS services, which include monitoring of layer 2 and 3 switches, routers, firewall, IDS and IPS.

2.4.1.4 Overlay

Cloud overlay network already discussed in detail in Section 2.4. The goal of overlay network monitoring is to improve efficiency and performance with end-to-end visibility of overlay network traffic in large cloud computing environment.

2.4.2 Cloud Monitoring studies Analysis

Few studies deal with the monitoring in cloud computing environment. These studies do not address in detail of low level, high level, underlay and overlay network monitoring metrics. For the monitoring in cloud computing environment, many other solutions proposed in the academic research. Each one of them has specific ability which might be a good choice, depending on the monitoring requirements.

2.4.2.1 Open Source Software for Cloud Monitoring

OpenNebula [43] is an open source management tool for heterogeneous cloud infrastructures. The functionality covered by OpenNebula as a cloud infrastructure

manager is extensive. It manages the physical resources, virtual machines, virtual networks and storage capacity. It also collects monitoring data via probes which have been installed on the systems. The main features are the provision of scalability and adaptability. OpenNebula has the ability to monitor high level, low level, underlay and virtual networks.

Nagios [44] is a well-known open source monitoring software that support monitoring of heterogeneous cloud computing environment. Nagios is based on the centralized client-server architecture to monitor cloud infrastructure. Nagios core architecture was designed for flexibility and scalability of monitoring. It provides several APIs to allow its feature-set to be easily extended through additional plugins. It is also used for monitoring OpenStack. Nagios have the capability to monitor high level, low level, underlay and virtual networks.

Nimbus [45] is an open source monitoring software that provides an efficient IaaS cloud monitoring solution. It allowed cloud monitoring for both consumer point of view and as well as cloud provider point of view. It is highly scalable monitoring software which could help the developers to customize monitoring according to users' requirements. Nimbus can monitor high level, low-level underlay networks.

Many other open source monitoring systems like GMonE [46], DARGOS [47], Lattice [5], PCMONS [48], mOSAIC [49] and CASViD [50] have been described previously.

2.4.2.2 Cloud Provider and Commercial Monitoring Software

Cloud providers have proprietary monitoring software that is available for consumers. It is provided according to the SLA defined metrics which could be used to monitor different layers of cloud. For example, Amazon Cloud Watch [51] could monitor applications running on AWS and Amazon EC2. AzureWatch [52] could monitor Azure-based resources including web applications, windows instances, SQL databases and windows storage. Licensed cloud monitoring software is available for monitoring several cloud platforms at once. For instance, Nimsoft [53] can be used to monitor Rackspace cloud, Google App Engine, Google Apps, S3 Web Services, Amazon EC2, Salesforce CRM and Microsoft Azure. Meanwhile, CloudKick [54] can be used to monitor Rackspace cloud, GoGrid and Amazon EC2. These tools have the abilities to monitor high-level, low-level and underlay networks metrics.

2.4.2.3 Related Research Work for Cloud Monitoring

In [55] S. Bardhan and D. Milojicic represent a prototype to monitor QoS measurement in a federated cloud environment. The author used a basic time-based mechanism to represent and measure QoS continuously for both individual service and composite services. However, the proposed prototype monitored only a single metric which is availability. The metric is based on servers' uptime and downtime that are hosted on different clouds. The limitation of the proposed mechanism is the parameter of the low-level monitoring, which considered only one metric.

Lee et al. [56] has proposed a service level QoS measurement for SOA environment. The author has discussed high level performance monitoring of web services and has provided mathematical formulas of different metrics of QoS measurement which include throughput, availability, accessibility and successability. However, it is still in the implementation of any proposed metrics.

Liu et al. proposed a lightweight system to monitor and to detect SLA violation in a cloud environment [57]. The proposed system was based on Simple Network Management Protocol (SNMP) [58] protocol. It monitored and obtained dynamic resource information for memory usage, CPU and link utilization for each VM for analyzing and detecting SLA violation. The proposed system only monitored some high-level parameters but lacking low level and overlay monitoring.

R. Grati, K. Boukadi, and H. Ben-Abdallah [59] proposed a framework for IaaS to SaaS monitoring of Business Process Execution Language that is processed in the cloud. The authors proposed manager and agent-based model for monitoring all cloud layers from both customer and cloud service provider point of view. However, the framework only monitored response time for low-level monitoring which was implemented in the given framework.

Alhamazani et al. [60] discussed the importance of dynamic monitoring of QoS in cloud layers. He determined how to adapt the unpredictable conditions is an inherently complex problem for which several technical implications must be considered in cloud environment. He described an ongoing Ph.D. work that developed methods to monitor the quality of service of cloud layers using SNMP [58]. His focus high level, low level and underlay network monitoring. Likewise, in

[24] the author propose cross-layer federated cloud application monitoring as a service framework. It is based on server and agent using standard protocol SNMP for data collection at all cloud layers or multi-cloud environment. The focuses are high level, low level and underlay network monitoring but not overlay network monitoring.

In [61], authors proposed SLA verification framework that leveraged a third-party auditor. They developed an effective testing algorithm that can monitor and detect SLA violations of the physical memory size of the virtual machine in cloud computing. They only focus on low-level monitoring and violation detection for VM memory size but no focus on other required parameters in cloud layers.

Shao et al. [21] proposed a runtime model based monitoring approach in a cloud environment. The author focused on basic cloud monitoring parameters in cloud layers. The model can organize cloud monitoring data which was collected by different methods and presented in a well-organized way. In addition, the presented model was capable of attaining a balance between runtime overhead and monitoring of cloud services. Generally, a high-level and low-level parameters were discussed but not implemented.

In [62], Adinolfi et al. propose a portable architecture for QoS monitoring in the cloud. He outlined the design and implementation of QoS-MONaaS framework. The downside of the framework was its dependable monitoring facility being the part of another project and only high-level monitoring for SaaS.

M. Dhingra, J. Lakshmi, and S. K. Nandy [63] proposed a distributed monitoring framework for cloud computing. They present a generic framework that could support customized monitoring using open source software. They focused on low-level monitoring in IaaS cloud layer but not high level and overlay network monitoring.

Kai et al. proposed SCM: a design and implementation of a monitoring system for CloudStack [64]. The authors presented a scalable and flexible monitoring system architecture to monitor the Apache CloudStack platform. The proposed prototype only monitored low-level parameters in IaaS cloud layer.

In [65], the authors presented features of different frameworks for cloud computing operating at IaaS level. They investigated the cloud monitoring process in main cloud platforms such as Eucalyptus [66], OpenNebula [43] and Nimbus [45]. The authors only observed low-level monitoring process and discussed the management challenges.

Lakshmanan et al. [10] proposed an outline that allows end-to-end monitoring of combined business applications. They developed and implemented a monitoring prototype to accomplish viewing of end-to-end monitoring data for various services. However, the focus was only on the high-level monitoring but no other monitoring type.

Katsaros et al. [67] proposed a cloud monitoring framework for measuring the quality of services in different cloud layers from application to infrastructure level. The proposed model was able to collect and combine monitoring data of runtime

resources allocation and decision making in cloud computing environment. The focuses were basic high-level and low-level monitoring but not overlay monitoring.

In [68], authors proposed a framework to design a monitoring model for simulation cloud. The focus was on high-level and low-level monitoring in a federated cloud environment that collect performance information metrics for cloud monitoring containing physical and virtual resources of the federated cloud. The proposed framework could detect real time anomalous behaviors of cloud resource consumption.

A. Meera and S. Swamynathan [69] proposed an agent-based resource monitoring system in IaaS cloud environment. The authors introduced the idea of how monitoring agents collect virtual machine resource usages like CPU and memory utilization. However, it was limited to low-level monitoring but not high level and overlay monitoring.

Katsaros et al. proposed cloud monitoring framework to monitor physical and virtual resources of the cloud [19]. Following REST architecture principle, it collected and stored monitoring information of physical and virtual resources of cloud infrastructure using open source software such as Nagios [44] and Libvirt [70]. The focuses were basic low-level, high-level and underlay network monitoring but not overlay network monitoring.

Smith et al. proposed an architecture for real-time monitoring in distributed cloud environment [71],. The proposed architecture has scalability and fault tolerant capability. Monitoring performance parameters were managed by streams that were

available to the users via push notifications. The focuses were high-level, low-level and underlay network monitoring but not overlay network monitoring.

Montes et al. [46] proposed an architecture for generic cloud monitoring in cloud computing environment. Author implemented the proposed cloud monitoring architecture which covered all three layers of cloud computing and it showed improved results. The author presented low-level, high-level and underlay network monitoring metrics but not overlay monitoring.

Povedano-Molina et al. proposed a distributed architecture for monitoring resources utilization in multi-clouds [47]. The proposed architecture could monitor different performance metrics for cloud monitoring including efficiency, reliability and multitenancy, without incurring a significant overhead. The monitoring involved low-level, high-level and underlay network monitoring but not overlay network monitoring.

Suciu et al. [72] proposed a model for network management and monitoring for cloud systems. He provided the way out to monitor cloud services using an open source monitoring tool named Nagios [44] and many other add-ons and plugins. The focuses were on low-level and underlay network monitoring but not overlay network monitoring.

In [73], authors proposed a scalable real-time monitoring architecture for large information systems with the goal for guaranteeing scalability and availability. The focus was the monitoring component of the logical cluster and physical racks for

large-scale network infrastructures hosted in data centers. It only monitored underlay network but not overlay monitoring.

Kim et at. [74] proposed a monitoring performance architecture for application services in multi-clouds. The key features of the architecture were flexible integration with external agents and separation of output channels such as real-time publish/subscribe and DBMS-based offline friendly repository for analysis of monitoring metrics. The focuses were low level and underlay network monitoring but not overlay network monitoring.

K. N. Mydhili and V. Gopalakrishna proposed a framework for monitoring cloud computing services [75]. They implemented a prototype for a few high-level monitoring parameters in SaaS layer using standard SNMP protocol. The drawback of the framework was lacking configuration management.

G. Liu and T Wood proposed a distributed software-based network monitoring framework for cloud data centers [76]. The authors presented a deployment strategy for a software-based packet monitoring system in the network to measure video contents using Software Defined Networking. Once data was captured, it was aggregated and sent to a processing engine. The focuses were underlay network monitoring but not overlay network monitoring.

Mann et al. [77] proposed a network service monitoring solution for cloud infrastructure. He identified problems within virtual switches for permitting flow-based network monitoring. They implemented and analyzed flow monitoring

protocols such as NetFlow [78] and sFlow [79] on physical switches and virtual switches in a cloud environment for underlay network monitoring.

L. Deri and F. Fusco [80] proposed a cloud monitoring architecture that described the real time traffic problems in heterogeneous cloud infrastructure. The proposed architecture was based on network probe, which was able to analyze classifying traffic into flows and to separate flows according to user requirements. Underlay monitoring information according to the user requirements was accessible via external software.

In [81], authors proposed a distributed monitoring system which collected the monitoring data using the Application Layer Traffic Optimization (ALTO) protocol [82]. They demonstrated that the ALTO protocol was very suitable to monitor the information for networked cloud infrastructure. They presented low level and underlay network monitoring but not overlay network monitoring in a cloud environment.

T. Mullins and A. Bagula [83] proposed a lightweight network management protocol for monitoring community clouds. It was an agent based monitoring tool using open source software SIGAR [84] that provided lightweight monitoring of community networks for cloud infrastructures. The proposed agent-based architecture was more flexible than the SNMP protocol in the case of data storage collected from the network, deployment and reduction of the traffic load in the network. The proposed prototype had low-level and underlay network monitoring capability but not overlay networks in the cloud.

M. Madan and M. Mathur proposed cloud network management model for monitoring cloud traffic [7]. The proposed model modified the traditional SNMP protocols to manage cloud traffic with more accurate results. They presented only underlay network monitoring but not high level, low level and overlay network monitoring for the cloud.

Clayman et al. [5] proposed a monitoring framework for resources utilities in virtualized cloud environments. The proposed framework could gather monitoring data such as memory, CPU and network utilization for each of the virtual execution environment within the cloud. The focuses were low level and underlay monitoring but not overlay monitoring.

Ward et al. proposed a monitoring system for a large cloud environment that is scalable and having ability to collect and analyze monitoring data for cloud infrastructure [6],. The proposed architecture described a technique for gathering the monitored parameters and also provided a mechanism to analyze these parameters in the large cloud. The focuses were low level and underlay network monitoring for IaaS layer.

P. Ya-Shiang and C. Yen-Cheng [25] proposed an SNMP protocol based on cloud virtual infrastructure monitoring framework. The authors introduced scalable SNMP agents to monitor VM resources in heterogeneous virtualized cloud environment. The proposed framework could manage monitoring resources according to the user requirements. They presented low level and underlay network monitoring in the cloud.

In [85], the authors proposed a prototype to monitor shared resource and throughput optimization for a cloud environment. They focus on low-level monitoring and discussed why shared cache contention is a serious problem in cloud virtualized environment and suggest a solution to them.

In [86], the authors proposed a lightweight framework for monitoring public cloud. The proposed design is less resource intensive and based on client-server architecture. They used open source plugins to monitor public cloud. A few high level and low-level monitoring metrics were discussed on SaaS and IaaS cloud layers.

In [48], authors proposed an architecture for private cloud monitoring resources. The proposed architecture was flexible and extensible. It can be adapted for centralize monitoring parameters to help improve QoS in a private cloud environment. They focused on low-level and underlay network monitoring on IaaS cloud layer.

J. Shao and Q. Wang [87] proposed performance guarantee approach for cloud applications based monitoring. They presented a cloud monitoring framework that collects real time data for PaaS metrics monitoring using a series of machine learning algorithms for future performance prediction in a cloud environment. They focused on high-level monitoring.

Emeakaroha et al. [50] proposed SaaS layer monitoring for SLA violation detection in cloud computing environments. He used standard protocol SNMP for efficient high-level monitoring and detection of SLA violation in SaaS cloud computing.

Rak et al. [49] proposed a cloud monitoring application using the mOSAIC framework. The proposed solution composed of four main components namely mOSAIC API, mOSAIC platform, mOSAIC provisioning system, and semantic engine. These components enabled the structure of custom monitoring systems for cloud monitoring using the mOSAIC API. The author also discussed high level, low level and underlay monitoring techniques using standard SNMP protocol to collect the data at different layers of cloud but lacking discussion on the implementation.

Table 2.1.

Comparison of cloud monitoring systems.

Research Work	**Cloud Layers**	**High Level**	**Low Level**	**Underlay**	**Overlay**
[9, 43, 55, 68]	SaaS	✓			
[80]	PaaS	✓			
[49]	SaaS, PaaS	✓			
[17, 60-61, 79]	SaaS, PaaS, IaaS	✓	✓		
[14-15, 20, 37, 40,42, 44, 53, 60]	SaaS, PaaS, IaaS	✓	✓	✓	
[5-6, 21, 56-57, 65, 67, 74, 76,]	IaaS		✓	✓	
[41, 48, 50, 52, 54, 58, 62, 78]	IaaS		✓		
[7, 66, 69-70, 73]	IaaS			✓	

From the above discussion and the comparison given in Table 2.1, it can be seen that monitoring gap in terms of overlay networks in a cloud environment. Unfortunately,

network monitoring features in above discussion systems are often limited, which means they can miss out on major performance issues in cloud environment. To keep a close watch on network and catch potential problems, need a network monitoring tool to track and report more in-depth on performance of overlay networks in a cloud environment.

2.5 Network Monitoring

Network monitoring primary focus is to initiate measurement tasks and collect raw data for measurement results, and provide the aggregated outcomes data for further analysis. The main concern is how to get required monitoring information in efficient way. In other words, a complete monitoring mechanism process that can guarantees components such as data repository, data aggregation, and data analysis can operate efficiently and interact seamlessly. Furthermore, an appropriate level of granularity for observation must be chosen while main approach for network management is how to maximize the overall performance, from the perspective of reliability, capability and latency, given that the network has to carry traffic for different applications, and different applications introduce different requirements and impacts on network performance.

2.5.1 Network Traffic Measurement Techniques

Network traffic measurement techniques typically have two types: Active and passive. Both measurements technique produces different form of information and the results do not essentially correlate well [88]. Furthermore, complete information of the network health can be gained by combining results from both active and

passive measurements called hybrid measurement technique. In this section traffic measurement techniques discuss in detail.

2.5.1.1 Active Measurement

Active measurements used to measure network traffic that are not possible by using passive measurements and they do not require high volume of storage space. Moreover, there are no privacy issues using active probing since the data used does not contain any private information. As all packets are generated artificially and on demand in active probe, therefore they usually contain only random bits as payload. Active measurement depends on injecting probe traffic with known condition into the network to test specific attributes of a network. The objective of probe packets is to present in detail that real network traffic is treated within the network [88].

In a typical way, an active measurement consists of two measurement sites to send and receive probe packets respectively. Moreover, through injecting probe packets at one end and retrieving the probe packets at the other end using active measurement technique can measure the properties of end-to-end network paths between sender and receiver.

Following active measurement tools generally used:

- **Ping**: use to send an ICMP ECHO packet to a receiver end and capture the ECHO reply packet to check connectivity to the receiver end and quickly measure the round trip time between the sender and the receiver.

- **Traceroute**: use to send a sequence of UDP probe packets with increasing time to live value of packet on the destination system, through receiving the ICMP time exceeded packet, identify the path along with intermediate nodes.

Followings are the benefits of active measurement technique:

- Measurements can be performed in due time.
- User constraints are not affected like security, privacy and access control problems.
- Measurements can be freely merged with the underlying network infrastructure with close to end user experiences.
- Measurement can be accomplished when passive measurement cannot be performed.

2.5.1.2 Passive Measurement

Passive measurements are typically use where capture points can be freely selected. This is best suitable for where single organization manage the whole owned network. This allows capture traffic to be any point on the path from the sender to the receiver [88].

Passive measurement does not require inject any traffic into the targeted system for measurement. It usually implements packet filters mechanism to capture basic traffic on the network and then using pattern-matching techniques searches for particular events. It is use libpcap for packet capturing which is available on most operating systems as common API, it helps to capture the type of packets and monitor the

specific interfaces at entry point of traffic. After the capturing of raw data packets for interested packets forwarded for further processing i.e interpretation of network protocols and parsing of packet header field as per requirement.

Following active measurement tools generally used:

- **Tcpdump** is an open source tool which capturing and displaying packet headers in detail, transmitted and received by Unix/Linux operating systems for monitoring, sniffing and collecting purpose.
- **NetFlow** is a propriety tool by cisco system and help to collect the flow data generated by enabled routers and switches for monitoring and data analysis purpose observed by a selected network interface.

Followings are the benefits of passive measurement technique:

- Usually a passive measurement required only one site to performed monitoring in the traffic path hence, it is easy to deploy.
- Use existing network resources for data collection that is already being processed by network devices like routers and switches.
- It is appropriate for specific monitoring and traffic measurements.

2.5.1.3 Hybrid Measurement

Hybrid measurement technique is a combination of active and passive measurements. i.e active probes are sent over a network and their progress is monitored by passive means during the measurement [89].

Followings are the benefits of hybrid measurement technique:

- Lower overhead and minimum impact on network traffic because no data store at network devices end.
- It provides high level of probability that the real user experience is being measured.
- Suitable for high speed and large network traffic analysis and performance monitoring.
- It is ideally suited as a scalable implementation and dynamic deployment.
- It is helpful to portray the complete picture of network behavior on traffic flows by adopting a service orientated approach.

2.6 Network Monitoring Techniques

Network monitoring is essential tool to ensuring that service provider is meeting user and application requirements according to the service level agreement. There are mainly three technologies used for network monitoring. In this section we have discussed about these monitoring techniques in detail.

2.6.1 Simple Network Management Protocol (SNMP)

SNMP is a component of the Internet Protocol suite as defined by the Internet Engineering Task Force (IETF) in RFC 3413 [90] . It is used mostly in network management systems to monitor and manage devices. Typically, these devices include servers, routers, switches, desktops etc. It is based on polling technology

which mean every monitoring device need to be configured for collection of the required monitoring data. It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects. However, three different SNMP versions introduces in the market till now. SNMPv1 and SNMPv2 almost have same messaging mechanism while in SNMPv3 also define user level security procedure. The key components of SNMP model are manager, agent and Management Information base (MIB). Figure 2.6 presents SNMP architecture.

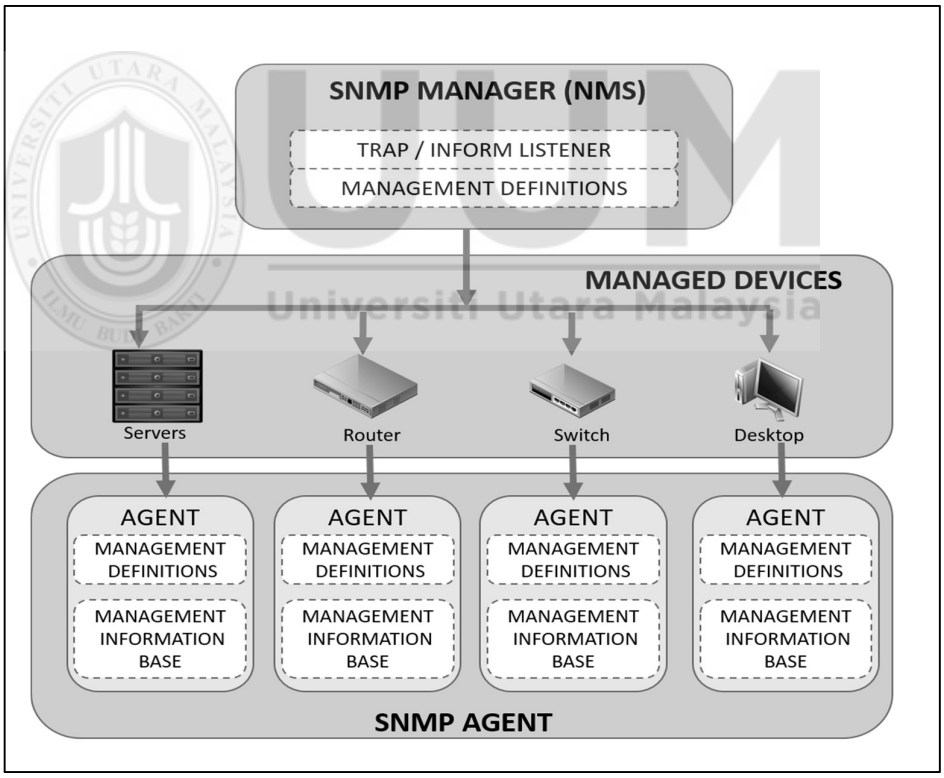

Figure 2.6. SNMP Architecture [91].

2.6.1.1 SNMP Manager

It is a software which is responsible to manage and monitor devices in the network using SNMP protocol. It can be act as a server that managing all of the configuration behalf of the users and network management application. It can send requests to different agents querying about required data which is based on management information base (MIB).

2.6.1.2 SNMP Agent

It is an application module that resides in a managed device. It is maintained the information about managed devices and responsible for answer all queries which is requested form SNMP manager using SNMP protocol.

2.6.1.3 Management Information Base (MIB)

It defines a set of variables which stored in the device and these variables managed using SNMP protocol.

2.6.1.4 SNMP Limitations

In SNMP we know that the Network Management Station (NMS) called manager periodically requests or polls the agent. The MIB inside agents contains a counter that counts number of bytes transmitted and received in a particular time interval on each of its interfaces. The counter is cyclic. The SNMP counters counts only a running total, and not the count the number of packets per interval. SNMP manager send polls to agent to compute packets per interval in short duration of time. SNMP has ability to poll after every five minutes. Thus SNMP poller periodically records

these counters and collects information [92]. SNMP has many known limitations to collection of monitoring data by polling technique. Some of limitations define bellow.

- Sometimes an SNMP poller restarts and it loses its track of a counter, counter resets (say after a router reboot), which results in large error in the estimate of traffic. In early versions of SNMP 32-bit counters were used and these counters reset quickly on high speed links. Sometimes SNMP poller wrongly calculates the average rate as per information received, ignoring the missing polls [26].
- "Jitter" caused by polling is another problem in SNMP. The Network Management Station must perform polls to many devices and these polls cannot be performed concurrently. These query –reply packets take some time to transit the network [27]. Finally, the result is that the reply packets reach late due to this jitter. Moreover, routers give low priority to SNMP packets; therefore they have a delayed response.
- SNMP processes on agents are given low-priority and hence they have a delayed response.
- SNMP is too periodic. Sometimes polling cycles from 30 seconds to several minutes long does not produce the actual picture of the network routing conditions. Even if we speed up the polling cycle it would miss many routing state changes, and would generate much management traffic overhead [26].
- SNMP communication delays the action to be taken by manager, as manager has to first send a query message in which it has to access the MIB , the object data then travel all the way to manager and if required send the update message to

manager. Thus, using SNMP is not meant for very large networks because sending a packet to get another packet causes delay in communication and hence in management. This type of polling causes large volumes of regular messages and end in problem response times that may be unacceptable [7].

- There is no acknowledgement for Trap messages in SNMP. If UDP is used with Internet Protocol (IP) to deliver trap message by agent, the agent gets no response whether the trap message has been delivered to manager or not. This is unacceptable for such critical messages.
- SNMP does not directly support crucial commands. The only way to prompt an event at an agent is indirectly by setting an object value. A more efficient way is to use remote procedure calls with parameters, conditions, status and results, that SNMP does not support.
- SNMP marginal errors should not be ignored as feeding such small errors into management process causes major problems, corrupting the results and leads to poor management and monitoring.

Furthermore, as the number of devices grow in the network, SNMP polling can generate significant amount of data in the network which create further load on the network. In addition, detailed troubleshooting and root-cause analysis of network performance is not possible with the level of data available through SNMP, so even if you know that a device has a problem, you cannot typically determine the exact nature of the problem in order to fix it. The above discussion provides detail information about SNMP protocol and its limitations. Therefore, it is not suitable protocol for high speed and large cloud network monitoring.

2.6.2 Packet Based Technology

Packet based technology used for detail packet inspection for troubleshooting and detail analysis of network traffic monitoring [93]. Every single data packet has to be captured including payload for further analysis to view the complete network activity. Packet level information required deep packet inspection and consume more computational resources [94]. However, packet based technology can deliver network traffic data same as SNMP and flow based technologies but with detail packet information for root cause analysis of network. Typically, it is used for intrusion detection system and prevention for network security threats. It required additional hardware to capture network traffic. Therefore, it is not suitable for high speed and large network traffic analysis where low computational resources required for real time traffic monitoring [95].

Table 2.2.
Comparison of monitoring technologies

Features	**SNMP**	**Packet based**	**Flow based**
Computational resources	low	high	**low**
Network traffic impact	medium	high	**low**
Data aggregation at source	no	no	**yes**
Extra hardware required	no	yes	**no**
Scalable for large networks	yes	no	**yes**
Layer 3 & above data	limited	yes	**yes**
High speed networks	no	no	**yes**
Push or pull access to traffic data	Pull	Push	**Push**

2.6.3 Flow Based Technology

A flow is sequences of packets stream with the set of common key fields between two end points of network devices. Typically, a flow is described by traffic that has the same key fields including source and destination IP address and ports, and protocol. If any of key field is changed, thus new flow will be created and if key fields are matched in the captured packet then it will be added to existing flow. Flow based technologies provides detail information of network performance information. Advantages of flow based techniques describe below.

- Use existing network resources for data collection that is already being processed by network devices like routers and switches.
- Aggregate information from different packets into flow records.
- It is based on push technology.
- Different level of traffic analysis possible.
- Lower overhead because no data store at network devices end.
- Volume of data is very less to export for further analysis
- Suitable for high speed and large network traffic analysis and performance monitoring.

There are few flow based sampling technologies around including Netflow [78], JFlow [96], sFlow [79] and IP Flow Information Export (IPFIX) [32]. All of these technologies have almost same flow sampling mechanism but different flow exporting and collection mechanism. NetFlow, JFlow and sFlow are proprietary

technologies [97]. NetFlow is Cisco Systems proprietary technology, JFlow is Juniper networks proprietary technology and sFlow is InMon corporation proprietary technology while IPFIX is only open source standard which is defined in RFC 7011 by IETF. The comparison of flow based technologies [98] given in table 2.3.

Table 2.3.
Comparison of Flow based technologies

NetFlow	**JFlow**	**sFlow**	**IPFIX**
Developed by Cisco Systems	Developed by Juniper Networks	Developed by InMon Corporation	**IETF Standard define in RFC 7011**
Proprietary	Proprietary	Proprietary	**Open Source**
Detail information for each flow	Detail information for each flow	Less information	**Detail information for each flow**
Sampling not 100% accurate	Sampling not 100% accurate	Sampling not 100% accurate	**1:1 packet sampling supported**
No payload	No payload	No payload	**No payload**
Layer 3 & above supported	Layer 3 & above supported	Layer 2 & above supported	**Layer 2 & above supported**
Limited scalability	Limited scalability	Scalable	**Scalable**

2.6.3.1 IP Flow Information Export (IPFIX)

The IPFIX protocol is not complex, which ensures simple implementation. IPFIX is only open source standard which is defined in RFC7011 by IETF. The IPFIX architecture of typical flow monitoring setups consists of several stages. Figure 2.7

presents IPFIX architecture design and processing stages in term of packet selection, flow processing and flow exporting.

The first stage is Packet Observation and packet capturing, in which packets are captured from an Observation Point. Observation Points can be line cards or interfaces of packet forwarding devices. The second stage is Flow Processing and Export, which consists of both a Flow Process and an Exporting Process. Within the Flow Process, packets are aggregated into flows, which are defined as "sets of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties" [32]. After a flow is considered to have terminated, a flow record is exported by the Exporting Process, meaning that the record is placed in a datagram of the deployed flow export protocol. Flow records are defined in [32] as "information about a specific flow that was observed at an observation point". The main task of the exporter is to take the measurement data from the flow cache and to send the IPFIX messages to data collectors.

The third stage is Flow Collector. Although it's not a part of IPFIX architecture but without this typical flow monitoring setup cannot completed. Its main tasks are working as flow reception and storage of flow data generated by the previous stage and the store data will use for analysis of traffic measurement.

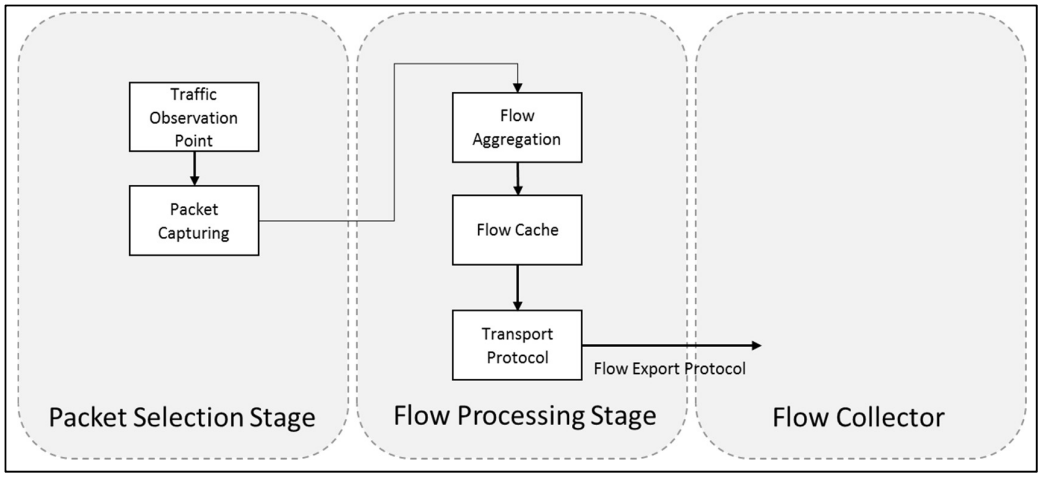

Figure 2.7. IPFIX Architecture [32].

Flow-based measurement is a popular method for various network monitoring usages. The sharing of flow-based information for monitoring applications having different requirements raises some open issues in terms of measurement system scalability, flow-based measurement flexibility, and export reliability [99]. Therefore, IP Flow Information Export (IPFIX) needs to enhance the functionality to help resolve these issues. As we discussed in detail section 2.4 and comparison given in Table 2.1 there is monitoring gap in terms of overlay networks in a cloud environment. Hence, we are going to enhance the IPFIX functionality and proposed enhanced IPFIX Framework for cloud overlay network monitoring.

2.7 Summary

This chapter mainly concentrated on a critical analysis of published literature in the

area of cloud monitoring. The main purpose of this literature review is to understand the work already carried out in the area that can help to highlight and identify the research gap. Furthermore, we have explained in detail related work and popular architectures of cloud network monitoring and finally we explained flow based methods for network monitoring and discuss in detail IP Flow Information Export protocol with their some open issues. The chapter summarizes the entire literature review under various properly selected sections and subsections for clarity. The research methodology adopted in this research work and the details of every step along with approaches used within those steps will be discussed in next Chapter Three.

CHAPTER THREE
RESEARCH METHODOLOGY

The primary objective of this research project is to come up with an enhanced IP Flow Information Export flow processing mechanism for cloud overlay network monitoring. The overall design of the mechanism is expected to have three sub mechanisms meeting the objectives presented in Chapter One. The mechanism developed as part of this project must be evaluated with a view of verifying and validating its operation to ensure that it achieved the objectives as mention Chapter One in 1.6. It is important that the research project has been carried out according to well defined plan, therefore the methods adopted at various stages of the project are scientifically valid and can be verified to produce similar results, if repeated under same conditions. This chapter presents the research methodology adopted in this work in order to active the said objectives in a valid and verifiable way.

The Design Research Methodology (DRM) [100] proposed by Blessing and Chakrabarti provides the basis for the methodology presented in this chapter. The proposed research methodology incorporates the modified or adopted stages of the DRM in order to make them better suited for this research project. The proposed research methodology consists of four phases as Analysis, Design, Testing and Evaluation as shown in Figure 3.1. The Design Research Methodology, in its basic form also consist of four stages and they are namely, research clarification and analysis, Descriptive Study-I, Prescriptive Studies and Descriptive Study-II. The

stages of the DRM along with the relationship between the stages, the means adopted at each stage and main outcomes shown in Figure 3.2.

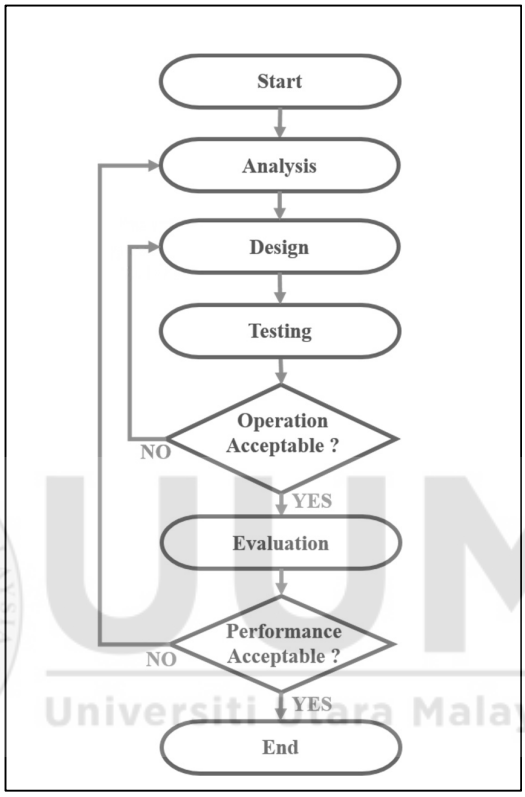

Figure 3.1. Research Methodology

The overall research approach adopted in this research project for achieving the research objective formulated in Chapter One. This section further explains in depth, how the DRM has been modified to suit the requirement of this project along with specific approaches employed in each stage along with the corresponding deliverables. Section 3.2 explain the first stage of the DRM methodology namely, Research Clarification along with the modifications that have been incorporated for the purpose of meeting the requirements of this

research project. The discussion centers on the aims of this stage, methods used in this stage and the deliverables. Section 3.3 present the adopted version of Descriptive Study I, the second stage of DRM with special emphasis on understanding the current status of technology, design and propose a conceptual model based on the gap identified. The methods adopted in designing of enhanced IP Flow Information Export flow processing mechanism for cloud overlay network monitoring presented in Section 3.4 under the Prescriptive Study phase of the DRM. Section 3.5 highlights the performance evaluation methods employed in this project along the discussion of metrics used. Finally, Section 3.6 concludes the chapter providing a comprehensive summary of the chapter.

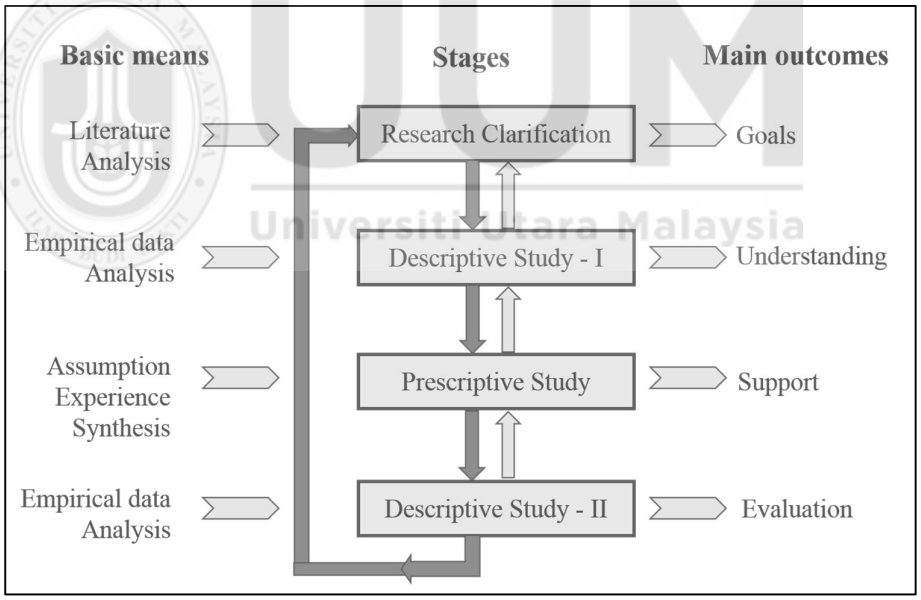

Figure 3.2. Research Methodology Stages [100].

3.1 Research Approach

The objective of this research project is to develop a monitoring system for cloud overlay networks based on IP Flow Information Export principals that can

monitor, identify and analyze the multi-tenant network traffic in high speed and complex cloud environment. Traditional cloud providers are struggling to keep up with new computing requirements. Network architects adopt simpler topologies and new control protocols to achieve better performance and operational agility in multi-tenant cloud networks. By adopting to the changes in the cloud network environment of service providers, the monitoring mechanism will furnish the accurate information to meet the demands of customers. The research approach adopted in this project must be scientific and comprehensive enough to guide the entire process from beginning to the end, so that the experiments conducted, and results produced are trustworthy and repeatable in comparable environment.

The methodology adopted in the research consists of four main phases. They are namely analysis, design, testing and verification and validation. In order to make the research approach scientifically strong and valid, it was decided to adopt features from the well established and scientifically proven DRM into the proposed research methodology [100, 101]. The DRM can be considered as an approach, guideline or a framework of supporting methods that enables the design research to be more rigorous, effective and efficient making the outcomes valuable both academically and practically [100]. The main objectives of DRM are as fellows.

- To provide a framework for researchers to conduct design research efficiently.
- To help researchers to identify research areas that are both academically and practically challenging and useful.

- To enable researchers to select and combine more than one research methods effectively.
- To provide guidelines for rigorous and systematic research planning.

Figure 3.3 shows the different phases of the research methodology adopted in this work in the light of DRM stages along with the methods used to achieve the objectives of each phase and the expected deliverables. The main phases of the research methodology are analysis, design, testing, and verification and validation. The corresponding DRM stages are research clarification, descriptive study I prescriptive study and descriptive study II. The process flows between the stages are shown in narrower arrows while the thicker arrows associate the methods used with the different stages of the DRM to the deliverables at each stage. The following sections provide brief explanations for each phase in conjunction with the explanation of methods and deliverables.

3.2 Analysis

The analysis phase of the research methodology is made up of the combination of both clarification and descriptive study I of DRM as shown ion Figure 3.3. This phase mainly concerns with obtaining a clear understanding of research project including the background, research problem, gap, questions and objectives culminating with the development of a conceptual model. The methods adopted at this phase include critical review of existing literature and experimental analysis

of existing mechanisms, in any. Further details of this phase are discussed in the first two stages of DRM.

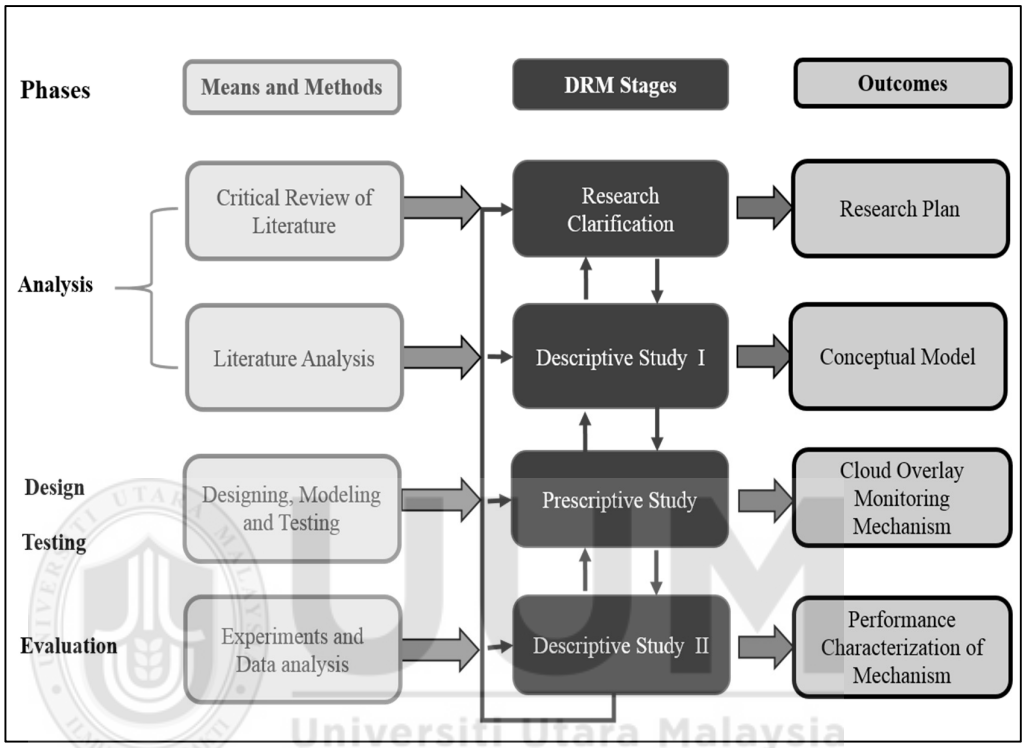

Figure 3.3. Research Approach

3.2.1 Research Clarification

It is necessary to obtain a clear understanding of the research to be conducted at beginning itself. By obtaining a clear and precise understanding of the current status of research are selected, it would be possible to define the goals of the research project with the challenging but realistic project plan. The research clarification stage can be further divided into six steps as shown in Figure 3.4.

These steps are inherently iterative helping to improve the results towards the objective gradually.

The method adopted at the research clarification stage is the analysis of the existing publish literature in the chosen area. At the end of this phase, the researcher would have acquired a broad and in depth knowledge in the chosen field especially the strengths and weaknesses of the current approaches and mechanisms. The weaknesses of the current approaches will be exploited to identify the research gap and then formulating research questions and objectives.

The main deliverable at the end of research clarification stage is Chapter One of the thesis. In addition to the main deliverable, a research survey paper could also be produced as subsidiary outcome. The research plan formulated at the conclusion of this stage would include the following.

- Research focus and motivation
- Research problem and questions
- Research objectives
- Area to be studied in depth
- Research approach including type, scope, important milestones, methods to be adopted and contributions.

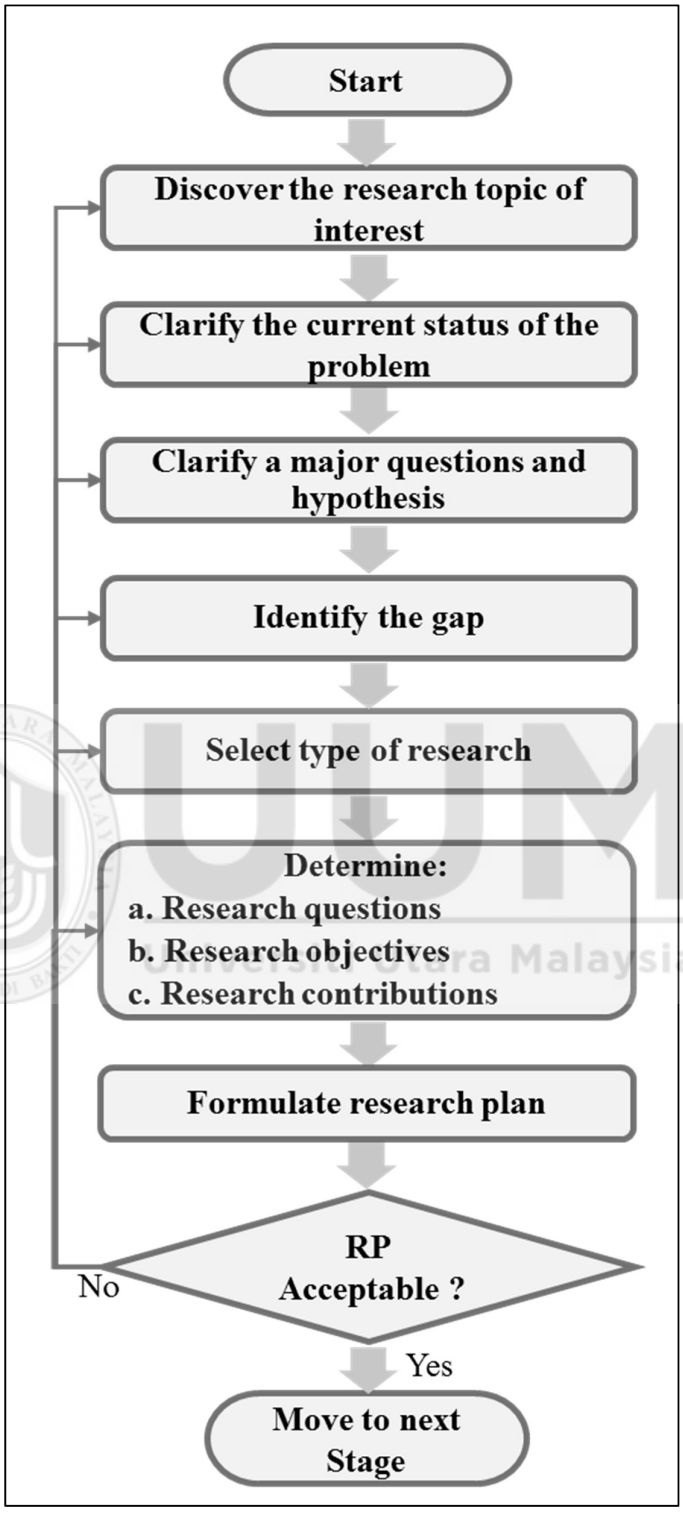

Figure 3.4. Steps involved in Research Clarification Stage

3.2.2 Descriptive Study –I

The Descriptive Study-1 makes the second part of the analysis phase of the research methodology. During the DS-1 stage, the research area identified in Research clarification stage would be further investigated with the view of obtaining an in depth understanding of the current status of the specific are selected. The method adopted at this stage would include critical review of literature along with analysis of existing approaches, where possible. During the course of this research an in depth study on the existing solutions was carried out [8]. DS-I is made up of five steps as shown in Figure 3.5, which must be applied recursively improving the understanding of the process further and deeper. The Conceptual model to be developed as part of DS-1 stage will also refined and improved at the application on each step as the deeper understanding of the shortcomings of the current mechanisms would work as catalyst for developing better solutions. The expected deliverables at the end of DS-1 include:

- Critical review of monitoring systems in high speed cloud network environment as presented in Chapter Two and
- Conceptual model for the proposed an enhanced IPFIX flow processing system for cloud overlay network monitoring.

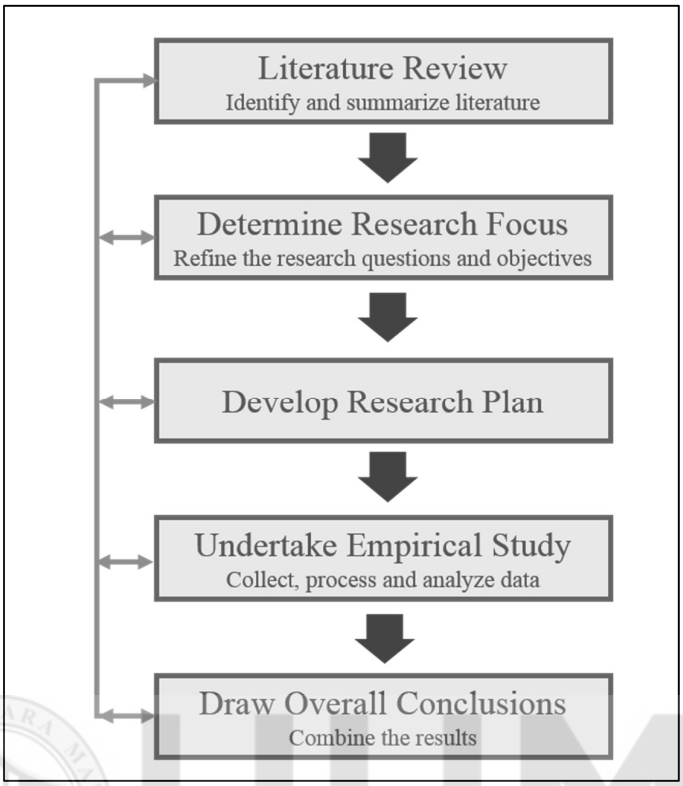

Figure 3.5. Steps involved in Descriptive Study –I

3.2.3 Conceptual Model

The cloud services are rapidly growing day by day due to fast adoption and migration of workload from private data centers to cloud data centers. A multi-tenant nature of cloud can be challenging for smooth management in terms of performance constraints and quality of service because the services of cloud are scalable, flexible and on demand. Since the cloud services are growing, network traffic also growing rapidly in cloud data centers. The complex cloud network infrastructure requires root cause analysis of network problems and troubleshooting in depth. To find the cause of problem may be searched several layers including physical and virtual layers that may be time consuming.

Therefore, a reliable and a real time monitoring system is required for both cloud provider and consumer to understand the performance issues and failure causes in cloud infrastructure [22]. In order to meet the requirement, the overall objective of this research has been formulated as mention in the section 1.6 to develop an enhanced IPFIX flow processing mechanism for cloud overlay network monitoring. Based on these objectives, a conceptual model of the research has been formulated. Figure 3.6 shows the conceptual model developed at the end of Descriptive Study-I.

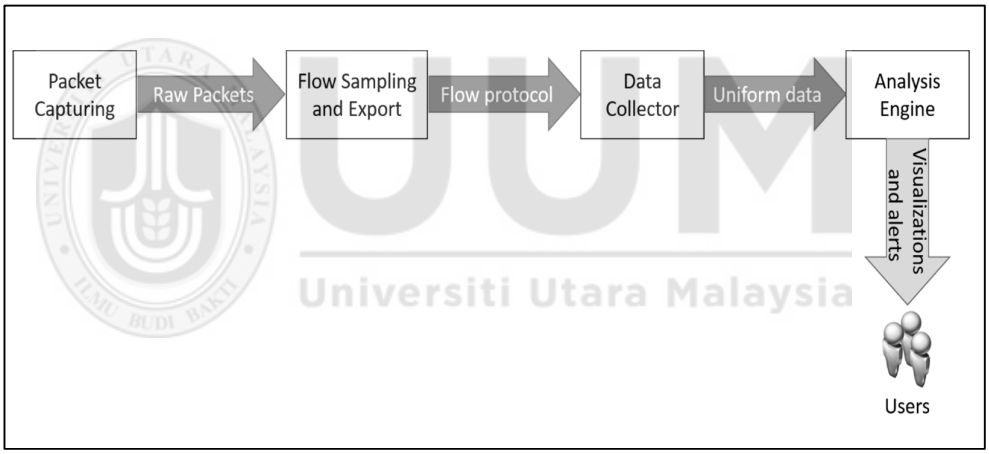

Figure 3.6. Conceptual Model for Network Traffic Monitoring Process

3.3 Design

This phase of research methodology aims to introduce the design of enhanced IPFIX framework which describes the whole process of packet capturing, flow generation, flow collection and data analysis. We also discuss in detail required components, implementation and testing of proposed framework. Figure 3.7 shows the steps involved in the mechanism development process.

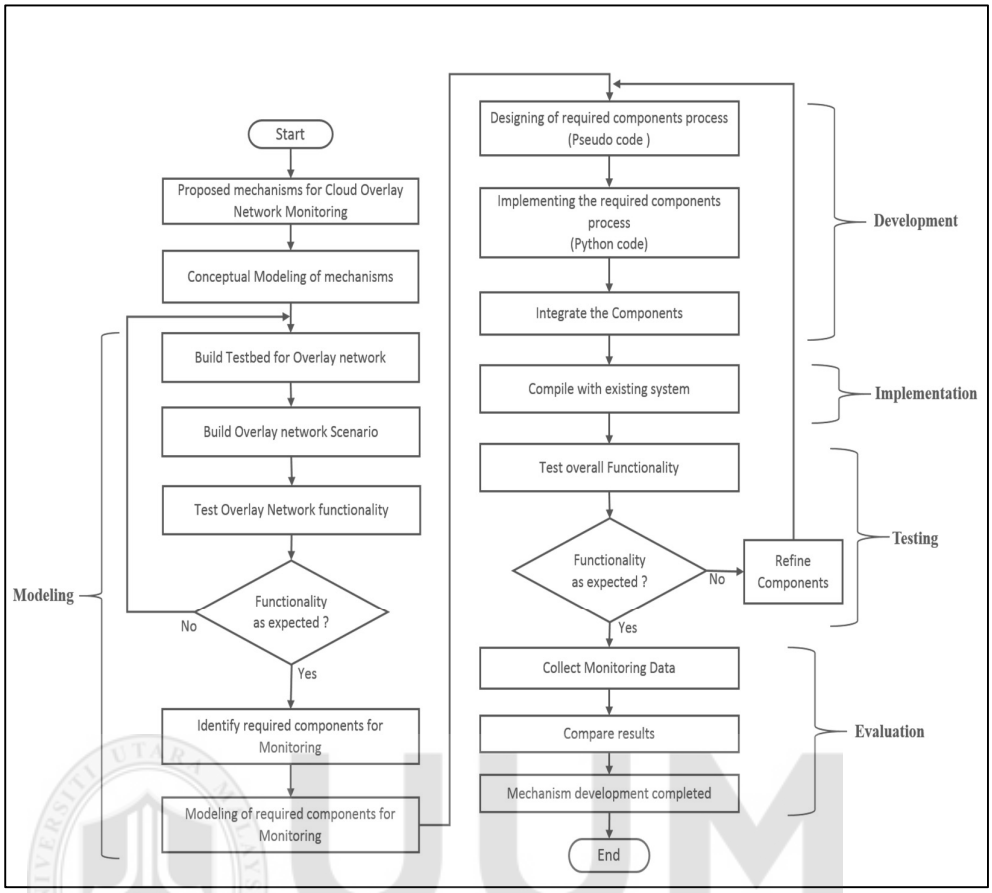

Figure 3.7. Mechanism Development Process

The main steps of this phase include the specification of the proposed mechanisms, modeling, development, implementation, testing and evaluation as shown in the Figure 3.7. The modeling and development process is generally a complex operation with many sub process and steps that must traversed many times iteratively. Many assumptions and simplifications may be made on this stage depending on the complexity of the model in order to arrive at realistic design. This also involves building the environment for simulation purpose. Implementation phase basically the integration of the sub process as single, large and self-contained unit but that can carry out a defined task independently and

testing for the overall functionality. This involves coding and compiling simulation environment. The evaluation and validation is the integration testing process that must be carried out to ensure the proposed mechanism fulfills the required objectives.

3.3.1 Design of Proposed Framework

The standard flow monitoring consists of three main layers. The first layer has two stages: Packet observation and selection stage, furthermore second layer has also two stages: flow aggregation and flow processing stage, while layer three used for flow data collection and analysis purpose. In this context packet selection and flow processing stages are often combined in a single device. Figure 3.8 presents the standard flow monitoring process stages.

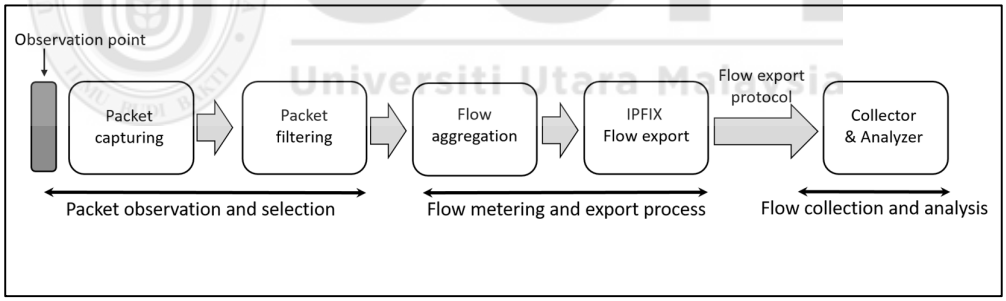

Figure 3.8. Standard Flow Monitoring Process

In large cloud data centers network, selection of traffic capturing techniques very important due to multi-tenant and dynamic nature of cloud. In high speed networks if packets are not capture properly due to any reason then it is difficult to analyze

accurate network performance issues. Packet capturing in virtual network environment similar to the physical network. Packet capturing task can be performed by very famous open source library named Packet capturing library (libpcap) [102]. This open source library gathers all packets in a raw form on the observation point and forwarded to further analysis. After collecting the pre-processing data which is form of captured packets will be aggregated into data flows and exported to the flow collector using flow based sampling technique. There are many flow based sampling standards around and they include: NetFlow, JFlow, sFlow and IP Flow Information Export (IPFIX). NetFlow, JFlow and sFlow are proprietary technologies while IPFIX is only open source and IETF standardized. For flow based sampling and export purpose we selected IPFIX standard which is described in in detail in RFC7011 [32]. The IPFIX support 1:1 packet sampling this mean considering every packet for data export and analysis. Furthermore, it is also support packet sampling and filtering mechanism as comparison given in table 2.3. To the selection of flow based sampling and flow exporter tool, it must be able to handle the IPFIX format data messages. Similarly, it should be support packet capturing mechanism and run on wide-spread operating system. For packet capturing, flow sampling and flow export purpose YAF-Yet Another Flowmeter [103] is selected as monitoring tool . Since it is an open source tool and its source code provides flexibility to configuring the parameters according to the requirements.

Table 3.1.

Comparison of IPFIX based Open Sources exporter

	Structured data RFC 6313	**Transport protocols**	**IANA supported Information Elements IEs**	**Option Templates supported**
nProbe		SCTP, TCP, UDP	✓	✓
nfdump		UDP	✓	✓
IPFIXcol		SCTP, TCP, UDP	✓	✓
Vermont		SCTP, TCP, UDP, file		✓
pmacct		UDP		✓
YAF	✓	SCTP, TCP, UDP, file	✓	✓

The comparison of IPFIX based open source exporter given in table 3.1.The proposed framework for enhanced IPFIX monitoring mechanism for cloud overlay network describes in Figure 3.9. As we know cloud overlay technologies introduces the same visibility challenges as most encapsulation methods. Essentially, end-to-end traffic is hidden inside the tunnel, so it must be able to strip away the encapsulation for sustained monitoring and troubleshooting. Therefore, the idea is our proposed framework to take advantage the flow based technique to capture the packets at observation point and reap the advantage of encoding and filtering technique to strip away the encapsulation layer for selection of cloud overlay packets. Finally, sampled packets will be aggregated into flows using proposed flow classification mechanism and message template mechanism, and exported to the collector for further data analysis. Flow collector

is responsible for collection of flow data which is exported by flow exporter and this is an essential part of flow monitoring system. It is working like a reception and received data from multiple flow exporters and store according to the requirement for further network traffic performance analysis[104]. Flow data generally does not contain any payload, the content of end user communications is protected. We used SiLk [105] as flow collector. SiLk understand IPFIX message sampled data and supported all transport protocols.

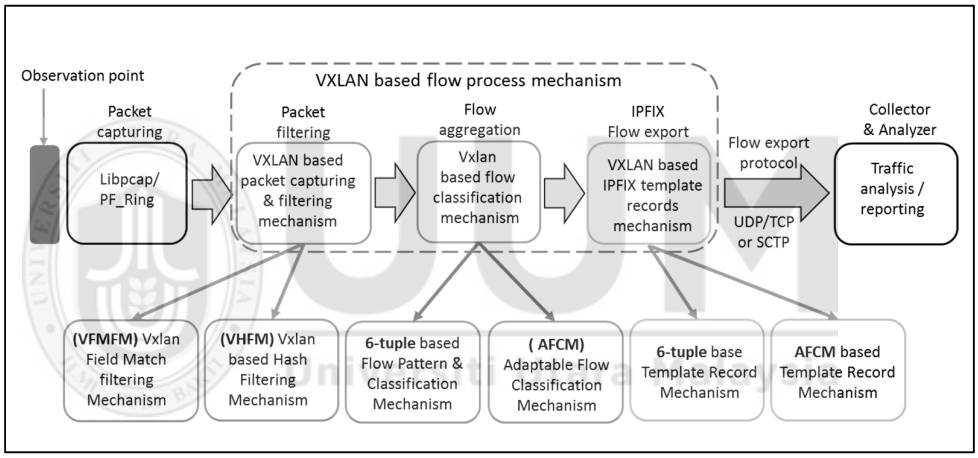

Figure 3.9. Proposed enhanced IPFIX flow processing mechanisms

This study intends to introduce enhanced IPFIX flow processing system which include packet filtering mechanism to capture overlay traffic, flow classification mechanism and VXLAN based data records for cloud overlay network monitoring purpose. Furthermore, we will develop plugin to above required components presented in Figure 3.9 for our proposed monitoring framework in order to achievethe research goals. While minimize the impact of computational

resources and reduce the bandwidth consumption, we use fast packet capturing, packet sampling and filtering techniques, and selection of flow records to increase the overall efficiency of our proposed monitoring framework.

3.3.2 Model Implementation

In the phase of implementation, we should take place all required components which we discuss in Section 3.3.1 for our proposed framework. This phase must be performed before we experiment our proposed model so that all components are working together correctly. Since this project is involved in the design of mechanism and techniques to monitor the cloud overlay networks in cloud environment. We need to build a cloud scenario for overlay network traffic. The implementation can be carried out on the experimental system in order to meet the requirements. Usually commercial systems come as single executable package that are ready for immediate installation and running. The developers of commercial systems provide only the minimum details of the inner workings of their software and no source code is made available to the general people. Hence it is necessary to go for an open source cloud systems where the entire source code along with the implementation details available to anyone without any limitations or license requirements. However, there is no simulator available to establish VXLAN based overlay network scenario for cloud network traffic. Due to lack of resources it is difficult to build a real time scenario. Therefore, it is challenging to build a VXLAN based cloud overlay network in order to run the

traffic in multi-tenant cloud environment and monitor the traffic as per our research requirement with minimum resources. Hence, we build a testbed scenario for cloud overlay network environment using Linux machines for implementation and testing of our proposed framework. Our proposed testbed comes with different software modules that when used together helps an organization build private and public cloud environment.

In order to provide a flexible virtualized data center service and implementation of overlay network, it is necessary to have an in-depth understanding of the implementation architecture of the system along with all the necessary tools. A tenant on an IaaS should be composed of not only virtual machines but also programmable networks which connect VMs with other networks through overlay network technologies. To build a cloud overlay network scenario consists of cloud controller and network controller mininet [106] emulator used on top of Open vSwitches (OVS) [107] on the hypervisors using OpenFlow [108] to establish overlay networks.

3.3.3 Model Validation

Model verification determines whether the model has been transformed from one from to another with sufficient accuracy [109]. In other words, the model verification process evaluates and verifies the accuracy of the porting of an application from a pseudocode or flowchart to an executable computer program developed in a high level language. It is also determining the degree of consistency of a computer model with respect to the real world phenomenon or

application [110]. Comparison with a real system is the most reliable and preferred way to validate a simulation model. When full measurement data is available it may be use trace-driven simulation to observe the model under exactly the same conditions as the real system. Assumptions, input values, output values, workloads, configurations and system behaviour should all be compared with those observed in the real world in order to ensure that the methods and mechanisms meet the intended requirements and the results obtained are valid [111].

3.4 Testing

The phase of testing we have to test our proposed framework to ensure that the functionality of our proposed framework is working properly and the results we received according to the requirements. Figure 3.6 presents the implementation scenario of overlay networks. After creating the scenario of cloud overlay networks which we will used to perform testing of our proposed framework. We will install network probes on different switches which we will used in our scenario for capturing the overlay network traffic on different interfaces in the cloud environment. After the installing of network probes we need to generate traffic between different virtual machines to measure the overlay network traffic. For this purpose we will use Iperf [112] to generate network traffic. Iperf is a benchmark and light user-level process that can be used to generate network traffic and testing between two hosts on a network and various types of aspect of networking performance. It is widely used and provides tests for both

unidirectional throughput and end-to-end latency either using TCP or UDP. Each virtual machine configured to its corresponding virtual machine for sending and receiving data for overlay network. Testing will be performed more than once for collection and measurement of overlay traffic. The collector that receives the data from network probes is configured on a dedicated machine. Same machine uses for analysis and visualization of captured data for cloud overlay networks.

3.5 Evaluation

Evaluation is the last phase of the research methodology adopted in this research. The main focus of this phase is the comprehensive evaluation of the mechanisms developed. Performance evaluation of cloud overlay network monitoring can be carried out using one of the three possible techniques as analytical modeling, simulation or testbed [99, 100, 101]. Evaluation phase encompasses the entire Descriptive Study-II stage of the DRM as shown in Figure 3.3.

3.5.1 Selecting the Evaluation Approach

The selection of the appropriate evaluation approach is significant step in any research project [116]. Therefore, different evaluation approaches have been compared for their strengths and weaknesses along with their suitability for the application in this work. However, hybrid technique has been selected as the main evaluation technique in this research project with combination of testbed and network simulation tools. Table 3.1 lists the strengths and weaknesses of each approach with respect to the application in computer network research.

Table 3.2.

Comparison of Different Evaluation Approaches

Criteria	**Analytical modeling**	**Simulation**	**Testbed**
Time Required	Low	Medium	High
Accuracy	Low	Moderate	Highest
Tool	None	Computer Software	Real Equipments
Trade off evaluation	Easy	Moderate	Difficult
Cost	Lowest	Moderate	Highest

3.5.1.1 Analytical Modeling

Analytical modeling or commonly known as mathematical modeling is a set of equations formulated using mathematical concepts to describe behaviour of a physical system [117]. The mathematical model thus developed can be used to analyze the performance of the system for various input conditions. The evaluation can be done either manually or automatically using a computer. Manual analysis may be cumbersome and take long time for some models depending on complexity of them. Using computers has become the common practice for solving analytical models due to the availability of advanced software packages equipped with libraries that can solve complex mathematical equations and function with minimal effort and custom configuration [118]. The results of an analytical model can be presented in many forms including tables and graphs [119]. This make visualization and comprehension of the result easier. It is possible to adjust the condition of the system varying the input or parameters with

relative ease. This method is very suitable for studying and provides an opportunity to obtain an initial view or understanding of a system before embarking on building a prototype for further study or the complete system.

The main shortcoming of this method is its difficult to build a complete system with mathematical models alone with the system becomes more complex with many interrelated units. Hence, when the complexity of a system increases it becomes necessary to make simplifications and assumptions to focus only on certain aspects of the system and to make the rest static. The advantages and disadvantages of analytical modeling include low cost, easier trade-off evaluation and low cost [120]. However analytical modeling accuracy is lower compared to other methods especially when the system becomes more complex.

3.5.1.2 Simulation

Simulation has been a widely used technique for analyzing the behaviour of dynamic system lately [121]. Simulation is carrying out experiments using computer based system models that have been developed using complex equations and algorithms. Simulation provides a flexible environment for carrying out in-depth study on protocols and mechanisms. The repeatability of experiments is very high with simulation. Also, with simulation, it is possible to study the effect of specific parameters on the output by changing only those parameters while keeping all the other constant. The enables the analysis of the performance of protocols and mechanisms in a scalable, controllable and repeatable environments [122]. Due to this reason, simulation has been widely

used in the performance evaluation and validation of results in computer networking research [100, 109].

3.5.1.3 Testbed

Testbed is the method of implementing a prototype of an actual network at lower complexity and scale with the objective of running tests and collecting results [124]. Commercial originations such as Intel Corporation and Hewlett Packard Enterprise have launched a cloud computing testbed spread across the United States, Singapore and Germany called Open Cirrus for conducting research in a real environment. But, access to this infrastructure is limited only to their member organizations. Testbeds can provide very accurate results as they mimic the real world scenarios using real equipment, but they are more difficult and expensive to construct[110, 111]. Building a cloud network environment with the minimum resources is a challenging task. Chapter Four explains how to build cloud network environment with minimum resources for cloud overlay network monitoring.

Hence a hybrid technique has been selected as the main evaluation technique in this research project with combination of testbed and network simulation tools. The main advantages of using hybrid technique for evaluation of result in research include their capability for efficient modeling and simulation of [93, 112, 113].

- Large scale cloud computing data centers and federated clouds
- Virtualized server hosts, with customizable policies for provisioning host resources to virtual machines
- Energy-aware computational resources

- Data center network topologies and message-passing applications
- Dynamic insertion of simulation elements
- User-defined polices for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

3.5.2 Evaluation Environment

The demand for tools for evaluation of network scenarios is rapidly growing because of the need to tests solution to network before it is used in the real world. Some of the factors including management complexity, cost, investment, benefit, and time required for implementation are some of the concerns associated with the today's cloud network environments with emerging network technologies. Cloud overlay is still a new and emerging technology and is being adopted in cloud network infrastructure, especially for virtual networking in the hypervisor for virtual machine to virtual machine communication. It is imperative to select the right tools to conduct the research evaluation technique in this project. Since there is no simulation suite available in the market to fulfil our cloud overlay network monitoring environment. Hence, a major challenge is how to build and simulate the cloud overlay network monitoring environment for obtained performance results of data transmission on multi-tenant cloud environment. These challenges include scaling to large networks, testing the correctness and evaluation the performance with the ability of easily migrate to real system with minimum changes for deployment. Due to this reason, a hybrid technique is designed, an emulator run on top of Linux testbed to evaluate this research project. There are many network simulator or emulator available in the market

with different features and capabilities. First we have to understand a difference between simulator and emulator, a simulator is designed to have a resemblance to the actual network operating system, but only simulate the specific function. On the other side, an emulator will run exact copy of an actual network operating system. Some of the popular network simulator or emulators include Cisco packet tracer [128], GNS 3 [129], EsitNet [130] and Mininet [127]. Each of these simulators has its own strengthen and weaknesses. A critical studies about these tools carried out in order to identify the right tool for this research.

Packet Tracer is a cross platform visual simulation tool designed by Cisco Systems that allows users to create network topologies and imitate modern computer networks. It is a powerful network simulation tool that allows users to experiment with network behavior. It provides simulation, visualization, authoring, assessment, and collaboration capabilities and facilities learning of complex technologies concepts. It can use to combine different network together to provide a high level overview of a network. However, this tool in research is limited as it lakes provision for new emerging network technologies for overlay networks.

GNS3 is a Graphical Network Simulator that allows the user to run multiple emulated system including different routers, Linux virtual machines and windows virtual machines. It allows to run different network operating systems in a virtual environment at single computer. GNS3 has a graphical front end tool called Dynagen. Dynamips is the core program that allows network operating system emulation. Dynagen runs on top of Dynamips to create a more user friendly, test

based environment. Users may create network topologies using simple windows ini type file with Dynafen running on top of Dynamips. GNS3 takes a step further by providing a graphical environment. Emulation is only possible for router and firewall platforms. Using an Etherswitch card in a router, switching platforms may also be emulated to the degree of the NIC card supported functionality. In GNS3, network simulation is not an easy task, especially if venture beyond a simple network topology as there is limited commands and parameters supported. This means GNS3 has limited functionality which is not supported new emerging technologies like flow controllers for virtual network environment.

EstiNet network simulator originates from Network and system laboratory at (National Chiao Tung University) NCTUns [130]. NCTUns had been used for network related research and publication since 2002. It became a commercial software on 2011 and was renamed EstiNet. It provides simulation environment including physical layer, media access layer, network layer, transportation layer and application layer. In addition, its GUI mode provides a convenient way to construct a simulated network and virtual display from simulation results observation and debugging. Linux kernel's TCP and UDP protocols stacks are directly integrated in simulated networks to provide layer 3 and layer 4 protocol behavior for network application. A Linux based network application program that can be run on real Linux network devices and can be run on a simulated network devices. It also supports flow based network simulation. Since it is commercial software therefore, it is not suitable for this research. Mininet [106] is an emulation tool that has been developed with the object of evaluating

different strategies, mechanisms and protocols. Table 3.2 shows a brief comparison of different network simulators and emulators available in the market.

Table 3.3.

Comparison of Different Network Simulators

General	Packet Tracer	GNS3	EstiNet	Mininet
Open source	No	Yes	No	Yes
Windows support	Yes	Yes	No	No
Linux Support	Yes	Yes	Yes	Yes
Emulation Mode	No	Yes	Yes	Yes
Compatible with real world controllers	No	No	Yes	Yes
Scalability	No	No	Yes	Yes
API supported	No	No	Yes	Yes

From table 3.2, it can be seen that Mininet would be the most suitable simulation tool for studying the different aspects of large network environment. Since it is an open source software, it can help to customizing the source code according to the requirement and design new emerging network protocols for high speed cloud network environment.

3.5.2.1 Mininet

Mininet was created by a group of professors at Stanford University to be used as tool to research and to teach network technologies [106]. The mininet open source network emulator is designed to support research and education in the field of Software Defined Networking system. Mininet is designed to easily create virtual network consisting of an OpenFlow controller, a flat Ethernet network of multiple OpenFlow enabled Ethernet switches, and multiple hosts connected to those switches. It has built in functions that support using different type of controllers and switches. It can create complex custom scenarios using Mininet Python API. It supports research, learning, development, prototyping, debugging and testing and any other task that cloud benefit from having a complete experimental network on a single computer.

It provides a simple and inexpensive network testbed for developing OpenFlow applications [131]. It has ability to run real code including standard Unix/Linux network applications as well as the real Linux kernel and network stack. The environment created in mininet for testing the code, including modified controllers, switches and hosts can move directly to real environment with minimum changes.

3.5.3 Experiment Steps

In order to carry out the experiment in a methodical manner and to confirm the repeatability of them, it is necessary to break or decompose the entire process into specific steps. It has been found that irrespective of the type of problem or

objective of the study the process used for the simulation study is constant. The simulation process is generally divided into eight basic steps [132]. The detail of steps are as follows:

i. The initial step involves clearly identifying the research problem to investigated along with defining the goals and objectives precisely. At this step, care must be taken to identifying and selected the most appropriate simulation tool to be used.

ii. Once the objectives are clear, the network topology to be simulated must be designed. The design should also include a suitable set of parameters real world scenarios.

iii. When the design is ready, the appropriate set of performance metrics must be identified for evaluating the performance of the network.

iv. In order to monitor the performance of the network, it is necessary to select the observation point on the network path for packet capturing process. In this step, the appropriate observation point selected.

v. At this step, the network topology designed in step 2 is implemented on the simulation tool selected.

vi. Following the topology implementation, it needs to be configured with right attributes for all components to reflect the selected scenario to be tested.

vii. Once everything is in order, the simulation program must be executed for the specific periods and the output data for selected output parameters must

be collected. It is necessary to conduct multiple simulation runs to make sure the results are free of bias.

viii. Finally, the required performance metrics need to be computed using the output data collected in Step vii and presented in the most appropriate format such as tables, graphs or chars along with the interpretation of them.

3.5.4 Performance metrics

Performance metrics is a set of attribute by researchers to measure and evaluate the operation of a specific equipment, algorithm, mechanism or protocol [133]. In order to monitor the overlay network and its performance, it is necessary that the users and service provider can access the performance metrics [134]. Thus, the deployment of an appropriate monitoring infrastructure and collection of specific measurement data becomes vital in any public network and cloud computing systems. These performance metrics must represent the expectations and requirement of both customers and providers. A set of Service Measurement Index (SMI) based on the International Organization for Standardization (ISO) standards for providing a standardized method for measuring and comparing cloud services have designed. The set of attributes developed are Accountability, Agility, Assurance of service, cost, performance, security and privacy and usability [135]. Out of these broad attributes, the performance related attributes or matrices have been selected in this research for developing the cloud overlay network monitoring mechanism.

It is necessary to mention here that in network performance measurement different metrics can be defined for different things in different contexts that can be used to characterize the derived values and could be define directly measured data. There are four types of metrics mostly attract attention of researchers in network monitoring that affect end-user application traffic. They are known as packet loss, packet delay, Processing load and throughput.

3.5.4.1 Packet Delay

In network communication, data move from one end to another end between sender and receiver and this data can be any form of file, audio and video etc. Practically, large data divided into small pieces and these small pieces of data transmitted to over the network and reassembled into original form of data at receiver end, these small pieces called packets. The time that these packets spent on the link from source to destination called latency. Generally, packet delay defined as the interval between the time when packet leave the source and passes through the link and reached the destination point. Often network links becomes overloaded due to so many packets processing and handle at once, therefore, packets take longer to get their destination and it is called high latency, in other words packet delay. There are various delays involve when packet passes through a network including routing delay, transmission delay and propagation delay.

Routing delay: a packet spent time inside a router for routing process and it can be further divided or broken down to queuing delay and packet processing delay.

Transmission delay: a time experienced by router when packet putting on the link.

Propagation delay: a time spent by a packet over the link.

All above mention factors involve as a result of packet delay. Packet delay may be measured as one-way delay or roundtrip delay.

The Packet delay can be calculated as follows:

Packet Delay = Propagation Delay + Serialization Dealy + Queuing Delay + Processing Delay

3.5.4.2 Packet Loss

In general packet loss defined as: when sender node sent out packets along with defined network path and they are not received by their destination as consequences of packet lost in transit. The main cause for packet losses are network congestion, overfull routing queues, errors in data transmission, unexpected outages, router reboots and routine maintenance. Since accurate information with respect to which packets are lost is hard to get and repeated lost might be have more severe impact than loss of single packet. In respect of real time audio or video applications, packet loss impacts the results and degradation of quality of services.

Packet loss is an important network performance metric and measured as a percentage of packets lost with respect to packets sent. First, we need to store

packet information into two variables TxPackets and RxPackets of two nodes then need to calculate loss ratio.

3.5.4.3 Processing Load

Besides the cloud overlay network performance related metrics, several additional metrics are used to explain the resources consumption of cloud overlay networks for examples CPU utilization and throughput. In order to measure the processing load on every step of enhanced IPFIX mechanisms, the results obtained from our proposed mechanisms are compared with standard IPFIX processing load. Furthermore, Average processing load is obtained on every stage of IPFIX, in order to compare with our proposed enhanced IPFIX mechanisms. Processing load is an important network monitoring performance matric and based on this measurement cloud provider and customers plan their future requirement of hardware capacity.

3.5.4.4 Throughput

Network throughput is generally referring to the maximum amount of data or message per time unit successfully deliver over a path or hop that can provide given existing utilization. It is measured in bits per second. The capacity of maximum network throughput is synonymous to communication link. During a given time interval on communication link, the remaining portion of the capacity that is not being used called available bandwidth. Network throughput is noticeably important monitoring the utilization of link resources and the link capacity. Maximum network throughput decided after the TCP three-way

handshake between source and destination and equal to TCP window size divided by the round-trip time of communications data packets.

The goal is to minimize the interruption of different overlay network performance issues and resources consumption in cloud network environment including: application network bandwidth utilization, locating and troubleshooting issues, applications and their effect on the network, workload utilization and unauthorized usage.

3.6 Summary

The chapter presented on the research methodology adopted in conducting this research. It consists of four phases namely, analysis, design, testing and evaluation. In order to arrive at scientifically valid and practically proven methodology, the stages of the designed research methodology have been adopted and integrated into different phases of this methodology. It has been proposed to develop enhanced IPFIX flow processing mechanism for cloud overlay network monitoring. The activities that will be carried out and the outcomes expected at the end of reaching each milestone were highlighted in this chapter in the light of different phases of the methodology and the stages of DRM. Phase one that includes both Research Clarification and Descriptive Study-I stages of DRM focuses on obtaining a clear understanding of the research. The main outcome of this phase includes identification of research problem, objectives and research questions, and development of conceptual model of this research. The design and testing phases of methodology concentrates on how to develop and test the

mechanisms for their functionality. The last activity in the proposed research is the evaluation of the mechanisms. This activity carried out through hybrid technique with combination of mininet network emulator tool on top of Linux based testbed for cloud overlay network monitoring. In the next Chapter, how cloud overlay network environment can be modelled for monitoring will be discussed in detail and using benchmark methodology performance of flow technologies will be presented in Chapter Four.

CHAPTER FOUR
PERFORMANCE OF FLOW TECHNOLOGIES WITHIN VXLAN ENVIRONMENT

This chapter presents the performance analysis of flow technologies with in VXLAN based cloud overlay network environment as mention in Chapter Three. Since VXLAN is new technology in cloud overlay network environment therefore, no simulation tool available to paradigm VXLAN based cloud overlay network environment to fulfil our monitoring requirement. Hence, it is imperative to select the supported components which are not only fulfil the requirements and also in the end to achieved the research goals of our proposed research. As the initial step at developing the environment, the required components were studied in detail to build a real time cloud overlay network environment. It has also been emphasized that the mechanism developed as part of this research must be extensible to single cloud environment to multi cloud environment. The organization of this chapter is as follows. Section 4.1 discused how to design and build VXLAN based cloud overlay network environment along with required components. Section 4.2 explains how to build virtual machines and virtual links for Cloud overlay network environment. Section 4.3 and 4.4 explains in detail development of VXLAN tunneling and VXLAN tunnel endpoints in virtual environment. Section 4.5 describes layer 3 routing for cloud overlay network environment. Section 4.6 presents the benchmark methodology to analysis of flow based technologies for our proposed environment. Finally, Section 4.7 summarizes the chapter highlighting the important points discussed in herein.

4.1 Building VXLAN Based Cloud Overlay Network Environment

In this phase we present to build a real time virtualized VXLAN based cloud overlay network environment. It is necessary to mention here the complete modeling of cloud overlay network environment is built at single computer machine with virtualized hypervisor and required necessary components. Prior to developing any virtualized modeling for the cloud based network, it is necessary to identify the required open source components which we can modify according to the requirements.

4.1.1 Required Components to build the Lab

Following is the list of components used to build a Lab environment.

- VMware workstation 12
- Linux Ubuntu Server 16.04 edition
- Open vSwitch 2.5
- Mininet 2.2
- Python
- SecureCRT
- X11 server

In order to build a cloud overlay network environment, first we need to build a Cloud underlay network. Therefore, we came up with a topology that could be easily demonstrate VXLAN based cloud network environment as presented in Figure 4.1.

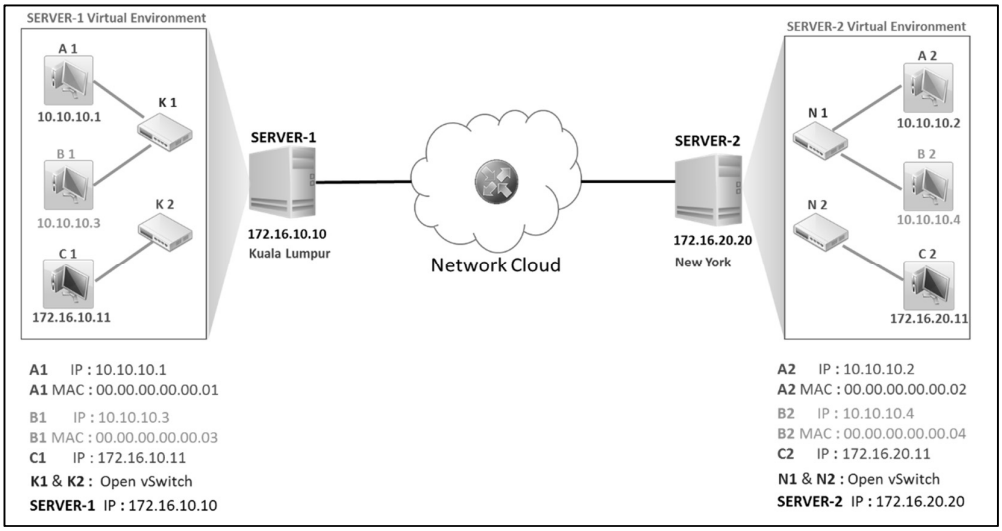

Figure 4.1. Cloud overlay network environment.

We have created three servers on virtualized hypervisor with different network segments on underlay networks, two servers SERVER-1 and SERVER-2 used for multi –tenancy environment for VM to VM communications respectively third server act as router to connect both SERVER-1 and SERVER-2 for underlay network communication to each other. For all the servers we installed Linux Ubuntu 16.04 Server edition with minimum packages only for cli mode as operating system. We have created two different IP network segments 172.16.10.0/24 for SERVER-1 (Kuala Lumpur) and 172.16.20.0/24 for SERVER-2 (New York) to connect with the SERVER-3 (Network Cloud) for routing purpose to each other underlay network communication. Figure 4.2 demonstrate the detail underlay network connectivity to all server machines.

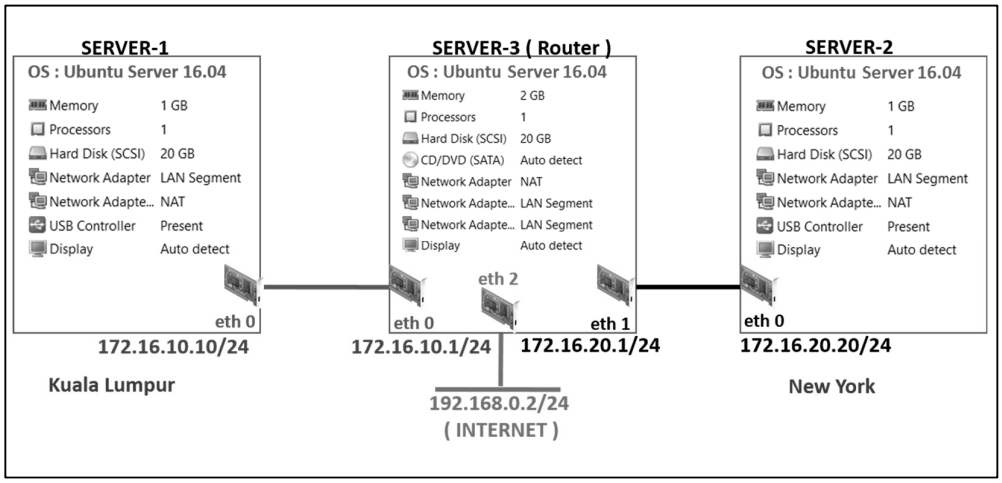

Figure 4.2. Cloud underlay network environment.

The SERVER-1 (Kuala Lumpur) LAN segment eth0 network IP : 172.16.10.10/24 connected to the SERVER-3 eth0 network IP : 172.16.10.1/24 and vice versa SERVER-2 (New York) Lan segment eth0 network IP : 172.16.20.20/24 connected to SERVER-3 eth1 network IP : 172.16.20.1/24. The SERVER-3 also connected to the WAN on eth2 network IP : 192.168.0.2/24 for internet and remote access purpose.

4.1.2 Modeling Cloud based Overlay Network

In order to build a VXLAN based cloud overlay network environment on the underlay network. We selected Open vSwtich [107] to build a VXLAN based cloud overlay network. Open vSwitch is the only open source switch which is supported VXLAN technology and IPFIX. It is also operated as a soft switch running within the hypervisor. On other hand to create a multi-tenant cloud environment where each VM isolated with other VM to keep the isolated communication. We need to

build virtual network and virtual machines within the environment to communicate with each other. Therefore, mininet [127] emulator selected to create instant virtual hosts, virtual network links, virtual switches and controller for flexible custom routing using openFlow [108]. It is also supported Open vSwitch within the hypervisor environment. Mininet is an open source emulator software and its virtual machines based on standard Linux network software.

We installed mininet with Open vSwitch on both servers SERVER-1 and SERVER-2 in order to create a virtual network environment for cloud overlay network. Furthermore, we also installed python scripting language in order to build a custom based virtual network topology using python script as mininet support python scrips. However, the minimum Linux kernel version 3.12 and Open vSwitch version 2.4 required to support VXLAN technologies and for IPFIX minimum Linux kernel version 3.10 required to build a VXLAN based cloud overlay network environment. Therefore, we installed latest Linux kernel version 4.4 and latest Open vSwitch version 2.5 on both servers. Figure 4.3 presents the detail version of software installed on both servers.

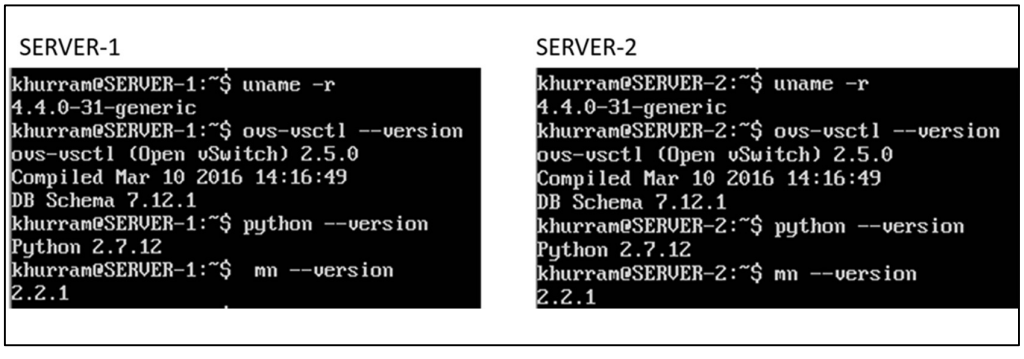

Figure 4.3. Detail of Installed software

4.2 Building Virtual Machines and Virtual links in mininet

To build a virtual machines and create virtual links to connect the Open vSwitch within this lab environment with mininet. We have the option of using mininet either create this lab environment manually or write a python script to build the lab environment within mininet. We decided to write a python script for each server to build this lab. Following are the components we used in mininet to build the lab environment as presented in Figure 4.1.

SERVER-1	SERVER-2
Virtual Machines (VM) o A1 o B1 o C1	Virtual Machines (VM) o A2 o B2 o C2
Open vSwitch (OVS) o K1 o K2	Open vSwitch (OVS) o N1 o N2
IP Address A1 – 10.0.0.1 B1 – 10.0.0.3 C1 - 172.16.10.11	IP Address o A1 – 10.0.0.2 o B1 – 10.0.0.4 o C2 - 172.16.20.11
Media Access Control (MAC address) A1 – 00:00:00:00:00:01 B1 – 00:00:00:00:00:03	Media Access Control (MAC address) o A2 – 00:00:00:00:00:02 o B2 – 00:00:00:00:00:04

Links	Links
A1 machine connect with K1	o A2 machine connect with N1
B1 machine connect with K1	o B2 machine connect with N1
C1 machine connect with K2	o C2 machine connect with N2

In order to match our lab topology we created two virtual machines on each server with hosts names A1, B1 and A2, B2, both hosts are connected with Open vSwitch K1 and N1 accordingly, every host assign a IP address and MAC address in order to easily understand the topology. Furthermore, we assigned same IP addresses and same MAC addresses on each hosts at SERVER-1 and SERVER-2 to easily understand how VXLAN technology works at cloud network environment. However, we created one other virtual machine with host name C1 connected with Open vSwitch K2 and N2, for normal communication to differentiate the VXLAN based and normal network traffic.

To test the mininet lab environment on both servers, we run following commands to ensure that host names, IP addresses and connectivity of hosts with the Open vSwitch in according to our lab requirements.

To check the host names and IP addresses, following is the output of mininet.

```
mininet> dump
<Host A1: A1-eth0:10.0.0.1 pid=934>
<Host B1: B1-eth0:10.0.0.3 pid=937>
<Host C1: C1-eth0:172.16.10.3 pid=940>
<OVSSwitch k1: lo:127.0.0.1,k1-eth1:None,k1-eth2:None pid=946>
<OVSSwitch k2: lo:127.0.0.1,k2-eth1:None pid=949>
mininet>
```

Figure 4.4. Detail of Server-1 dump output in mininet

```
mininet> dump
<Host A2: A2-eth0:10.0.0.2 pid=936>
<Host B2: B2-eth0:10.0.0.4 pid=939>
<Host C2: C2-eth0:172.16.20.3 pid=942>
<OVSSwitch n1: lo:127.0.0.1,n1-eth1:None,n1-eth2:None pid=948>
<OVSSwitch n2: lo:127.0.0.1,n2-eth1:None pid=951>
mininet>
```

Figure 4.5. Detail of Server-2 dump output in mininet

To check the connectivity links between hosts and switch, following is output on mininet.

```
mininet> net
A1 A1-eth0:k1-eth1
B1 B1-eth0:k1-eth2
C1 C1-eth0:k2-eth1
k1 lo:  k1-eth1:A1-eth0 k1-eth2:B1-eth0
k2 lo:  k2-eth1:C1-eth0
mininet>
```

Figure 4.6. Detail of Server-1 connectivity links between hosts and switch


```
mininet> net
A2 A2-eth0:n1-eth1
B2 B2-eth0:n1-eth2
C2 C2-eth0:n2-eth1
n1 lo:  n1-eth1:A2-eth0 n1-eth2:B2-eth0
n2 lo:  n2-eth1:C2-eth0
mininet>
```

Figure 4.7. Detail of Server-2 connectivity links between hosts and switch

A1 host using A1-eth0 network interface connected to switch K1 on port k1-eth1 and B1 host network interface connected with same switch K1 on port k1-eth2. Similarly A2 host using A2-eth0 network interface connected to switch N1 on port n1-eth1 and B2 host network interface connected with same switch N1 on

port n1-eth2. However, C1 and C2 hosts network interfaces connected with different virtual switches K2 on port k2-eth1 and N2 on port n2-eth1.

There is one bridge on each Open vSwitch with one physical interface and two virtual ports. Following is the output of Open vSwitch on mininet.

```
mininet> sh ovs-vsctl show
ba1bc60a-aaba-4c21-b058-ef6970eb53c3
    Bridge "k1"
        fail_mode: secure
        Port "k1-eth1"
            Interface "k1-eth1"
        Port "k1"
            Interface "k1"
                type: internal
        Port "k1-eth2"
            Interface "k1-eth2"
    Bridge "k2"
        fail_mode: secure
        Port "k2-eth1"
            Interface "k2-eth1"
        Port "k2"
            Interface "k2"
                type: internal
    ovs_version: "2.5.5"
mininet>
```

Figure 4.8. Virtual bridge detail on Server-1

```
mininet> sh ovs-vsctl show
ba1bc60a-aaba-4c21-b058-ef6970eb53c3
    Bridge "n2"
        fail_mode: secure
        Port "n2-eth1"
            Interface "n2-eth1"
        Port "n2"
            Interface "n2"
                type: internal
    Bridge "n1"
        fail_mode: secure
        Port "n1"
            Interface "n1"
                type: internal
        Port "n1-eth1"
            Interface "n1-eth1"
        Port "n1-eth2"
            Interface "n1-eth2"
    ovs_version: "2.5.5"
mininet>
```

Figure 4.9 Virtual bridge detail on Server-2

4.3 VXLAN Tunneling

To achieve connectivity between isolated networks that needed to share the same policies. We need to build an overlay network on underlay network, where tunnels are created between the hypervisors in different locations allowing virtual machines to be provisioned independently from the physical network. Following is the mechanism to create an overlay virtual Layer 2 network on top of the physical Layer 3, based on VXLAN tunnels between Open vSwitch bridges running on separate machines SERVER-1 and SERVER-2. Following configuration added on Open vSwitchs on mininet at both servers.

SERVER-1:

```
sh ovs-vsctl add-port k1 vtep -- set interface vtep type=vxlan
option:remote_ip=172.16.20.20 option:key=flow ofport_request=10
```


SERVER-2:

```
sh ovs-vsctl add-port n1 vtep -- set interface vtep type=vxlan
option:remote_ip=172.16.10.10 option:key=flow ofport_request=10
```

We added VXLAN Tunnel Endpoint (VTEP) services on both Open vSwitchs to communicate with each other remotely and also added flow forwarding port functionality using OpenFlow controller.

4.4 VXLAN Tunnel Endpoint (VTEP)

Frame encapsulation is done by an entity known as a VXLAN Tunnel Endpoint (VTEP.) A VTEP has two logical interfaces: an uplink and a downlink. The uplink is responsible for receiving VXLAN frames and acts as a tunnel endpoint with an IP address used for routing VXLAN encapsulated frames. These IP addresses are infrastructure addresses and are separated from the tenant IP addressing for the nodes using the VXLAN. The source and destination IP addresses used for VXLAN are the Source VTEP and destination VTEP. This means that the VTEP must know the destination VTEP in order to encapsulate the frame.

Furthermore, this lab setup is totally proactive which means we can either follow the controller entries or we can create manually entries to complete the lab setup. Due to lack of hardware resources we decided to add manual flow entries with VXLAN Network Identifier (VNI) to differentiate the individual VXLAN overlay network traffic for virtual machines communications. Following are the flow entries of controller we added manually in order to VXLAN based tunneling communication using python script.

```
table=0,in_port=1,actions=set_field:100->tun_id,resubmit(,1)
table=0,in_port=2,actions=set_field:200->tun_id,resubmit(,1)
table=0,actions=resubmit(,1)
table=1,tun_id=100,dl_dst=00:00:00:00:00:01,actions=output:1
table=1,tun_id=200,dl_dst=00:00:00:00:00:03,actions=output:2
table=1,tun_id=100,dl_dst=00:00:00:00:00:02,actions=output:10
table=1,tun_id=200,dl_dst=00:00:00:00:00:04,actions=output:10
table=1,tun_id=100,arp,nw_dst=10.0.0.1,actions=output:1
table=1,tun_id=200,arp,nw_dst=10.0.0.3,actions=output:2
table=1,tun_id=100,arp,nw_dst=10.0.0.2,actions=output:10
table=1,tun_id=200,arp,nw_dst=10.0.0.4,actions=output:10
table=1,priority=100,actions=drop
```

Figure 4.10. Flow entries for Overlay network communication on Server-1

```
table=0,in_port=1,actions=set_field:100->tun_id,resubmit(,1)
table=0,in_port=2,actions=set_field:200->tun_id,resubmit(,1)
table=0,actions=resubmit(,1)
table=1,tun_id=100,dl_dst=00:00:00:00:00:02,actions=output:1
table=1,tun_id=200,dl_dst=00:00:00:00:00:04,actions=output:2
table=1,tun_id=100,dl_dst=00:00:00:00:00:01,actions=output:10
table=1,tun_id=200,dl_dst=00:00:00:00:00:03,actions=output:10
table=1,tun_id=100,arp,nw_dst=10.0.0.2,actions=output:1
table=1,tun_id=200,arp,nw_dst=10.0.0.4,actions=output:2
table=1,tun_id=100,arp,nw_dst=10.0.0.1,actions=output:10
table=1,tun_id=200,arp,nw_dst=10.0.0.3,actions=output:10
table=1,priority=100,actions=drop
```

Figure 4.11. Flow entries for Overlay network communication on Server-2

4.5 Layer 3 Routing For Cloud Environment

To communicate between different LAN segments SERVER-1 and SERVER-2 we need layer 3 routing functionality as presented in Figure 4.2. Therefore, we added SERVER-3 between both servers SERVER-1 and SERVER-2. These both servers not connected directly to each other, they are connected via SERVER-3. Thus, we added routing functionality on SERVER-3 in order to network communication between SERVER-1 and SERVER-2. To do this we enable the IP forwarding and add the manual routes to the SERVER-3. Following is detail of routing table of SERVER-3.

khurram@ubuntu:~$ route –n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use         Iface
172.16.10.0     172.16.10.1     255.255.255.0      UG    0         0         0   eth0
172.16.10.0     0.0.0.0         255.255.255.0      U     0         0         0   eth0
172.16.20.0     172.16.20.1     255.255.255.0      UG    0         0         0   eth1
172.16.20.0     0.0.0.0         255.255.255.0      U     0         0         0   eth1
```

Figure 4.12. Routing table entries on Server-3

After adding the routes on routing table, physical network communication enables between SERVER-1 and SERVER-2. To test the VXLAN based cloud overlay network lab setup environment we used ping utilities to generate the traffic from virtual machine A1 to virtual machine A2 network communication. Following is the output of mininet at SERVER-1.

```
mininet> A1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.732 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.106 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.059 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.052/0.201/0.732/0.266 ms
mininet>
```

Figure 4.13. Output results of A1 ping

```
mininet> B1 ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.053 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.086 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.092 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.091 ms
^C
--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.053/0.082/0.092/0.018 ms
mininet>
```

Figure 4.14. Output results of B1 ping

Furthermore, we used tcpdump to ensure that communication working properly on VXLAN based cloud overlay network. Following are the output of tcpdump on SERVER-1.

khurram@SERVER-1:~$ sudo tcpdump -i eth0

```
khurram@SERVER-1:~$ sudo tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:02:47.815860 IP 172.16.10.10.52354 > 172.16.20.20.4789: VXLAN, flags [I] (0x08), vni 200
IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 4666, seq 124, length 64
22:02:47.816517 IP 172.16.10.10.52354 > 172.16.20.20.4789: VXLAN, flags [I] (0x08), vni 100
IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 4667, seq 120, length 64
22:02:47.817193 IP 172.16.20.20.41975 > 172.16.10.10.4789: VXLAN, flags [I] (0x08), vni 200
IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 4666, seq 124, length 64
22:02:47.817227 IP 172.16.20.20.41975 > 172.16.10.10.4789: VXLAN, flags [I] (0x08), vni 100
IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 4667, seq 120, length 64
22:02:48.818680 IP 172.16.10.10.52354 > 172.16.20.20.4789: VXLAN, flags [I] (0x08), vni 200
IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 4666, seq 125, length 64
22:02:48.819330 IP 172.16.10.10.52354 > 172.16.20.20.4789: VXLAN, flags [I] (0x08), vni 100
IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 4667, seq 121, length 64
22:02:48.819930 IP 172.16.20.20.41975 > 172.16.10.10.4789: VXLAN, flags [I] (0x08), vni 200
IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 4666, seq 125, length 64
22:02:48.819959 IP 172.16.20.20.41975 > 172.16.10.10.4789: VXLAN, flags [I] (0x08), vni 100
IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 4667, seq 121, length 64
^C
8 packets captured
8 packets received by filter
0 packets dropped by kernel
```

Figure 4.15. Output of tcpdump on SERVER-1

Using the above output of tcpdump VXLAN frames are sent to the IP address assigned to the destination VTEP.

A1 → A2

ICMP echo request	ICMP echo reply
Source IP : 172.16.10.10	Source IP : 172.16.20.20
Source port : 52354	Source port : 41975
Destination IP : 172.16.20.20	Destination IP : 172.16.10.10
Destination port : 4789	Destination port : 4789
VNI :100	VNI :100
VM A1 IP : 10.0.0.1	VM A2 IP : 10.0.0.2
VM A2 IP : 10.0.0.2	VM A1 IP : 10.0.0.1

B1 → B2

ICMP echo request	ICMP echo reply
Source IP : 172.16.10.10	Source IP : 172.16.20.20
Source port : 52354	Source port : 41975
Destination IP : 172.16.20.20	Destination IP : 172.16.10.10
Destination port : 4789	Destination port : 4789
VNI :200	VNI :200
VM B1 IP : 10.0.0.3	VM B2 IP : 10.0.0.4
VM B2 IP : 10.0.0.4	VM B1 IP : 10.0.0.3

It is necessary to mention here this lab is extensible from single cloud to multi cloud environment as we are connected with different LAN segments. The goal is here to provide the real time environment for VXLAN based cloud overlay network to fulfil the research requirements.

4.6 Analysis of Flow based Technologies within VXLAN Environment

The virtualization plays a vital role to implement cloud computing, However, virtualization technologies added an additional level of complexity. In cloud overlay network environment, VXLAN introduces extra CPU workload on each packet processing at hypervisor level in order to encapsulate and decapsulate packets on sender and receiver end. Hence, it is imperative to ensure the performance evaluation of proposed VXLAN based cloud overlay network environment. Besides the cloud overlay network performance related metrics, several additional metrics are used to explain the resources consumption of cloud

overlay networks for examples CPU utilization, throughput and latency. Therefore, we performed experiments with flow based technologies including NetFlow[30], sFlow[31] and IPFIX [136] in order to define the benchmarking of our proposed VXLAN based cloud overlay network environment. The de facto methodology for benchmarking of network performance is defined in RFC 2544 [137] by the Internet Engineering Task Force (IETF) . The RFC provides an out of service benchmarking methodology using throughput and packet loss tests to validate the network performance results. The methodology defines the byte size to run the test duration and number of test iterations.

To test the different flow technologies in our proposed VXLAN based cloud network environment. We installed network probes on Server-3 working as a router, which is used in our scenario for capturing the network traffic on interface and measurement of CPU workload. After installing of network probes, we generated traffic between different virtual machines to measure the overlay network traffic. For this purpose we used iperf [112] to generate network traffic, iperf is a light user-level process that can used to generate network traffic and testing between two hosts on a network and various types of aspect of networking performance. It is widely used and provides tests for both unidirectional throughput and end-to-end latency either using TCP or UDP. As per research requirement, simulation performed to evaluate the performance with different flow technologies, refer to the picture 4.1 configuration, traffic generated between different network segments using benchmarking methodology with transmission duration 60 minutes based on given datasets in Table 4.1.

Table 4.1.

Data sets with traffic transmission rates for simulation

Transmission duration: 60 minutes	**Data Transmission rate for 64 kps dataset**	**Data Transmission rate for 1 Mbps dataset**
VM-A1 → VM-A2	21 Kbps	400 Kbps
VM-B1 ← VM-B2	25 Kbps	250 Kbps
VM-C1 → VM-C2	18 Kbps	324 Kbps
Total	**64 Kbps**	**1 Mbps**

Each virtual machine configured to its corresponding virtual machine for sending and receiving data for overlay network. Testing has been performed more than once for collection and measurement of overlay traffic and processing load.

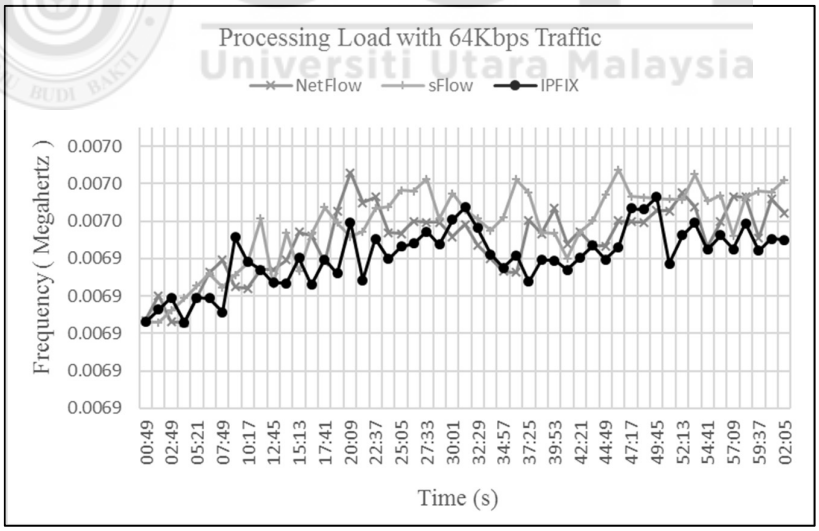

Figure 4.16. Processing load analysis of different network probes with 64 kbps traffic

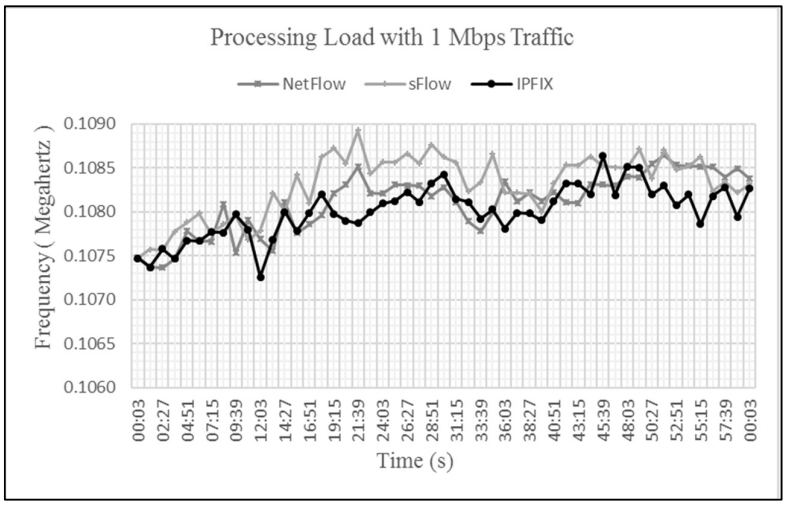

Figure 4.17. Processing load analysis of different network probes with 1 Mbps traffic.

Figure 4.16 shows different flow monitoring technologies network probe processing load analysis with 64 Kbps traffic between Netflow, sFlow and IPFIX at observation point. Similarly Figure 4.17 presents same network probes processing load analysis of 1 Mbps traffic. However, it can be seen that standard IPFIX took slightly less processing load as compare to NetFlow and sFlow technologies. While it is imperative to ensure that results are accurate, number of tests performed with same environment. Table 4.2 presents 64 Kbps traffic load and Table 4.3 presents 1 Mbps traffic load data, collected during experiment performed on overlay network environment.

Table 4.2.

Average processing load collected in MHz during 64 Kbps traffic transmission

No of test	NetFlow	sFlow	IPFIX
1	0.006939828	0.006947495	0.00693116
2	0.006963242	0.006963230	0.00694648
3	0.006950042	0.006976324	0.00693989
4	0.006954817	0.006955584	0.00694639
5	0.006978775	0.006972395	0.00696399
6	0.006963242	0.006967463	0.00695271
7	0.006971392	0.006971558	0.00696565
8	0.006963242	0.006972574	0.00695303
9	0.006967973	0.006964888	0.00695897
10	0.006963242	0.006963832	0.00697110
Avg. Load	**0.006961579**	**0.006965534**	**0.006952937**

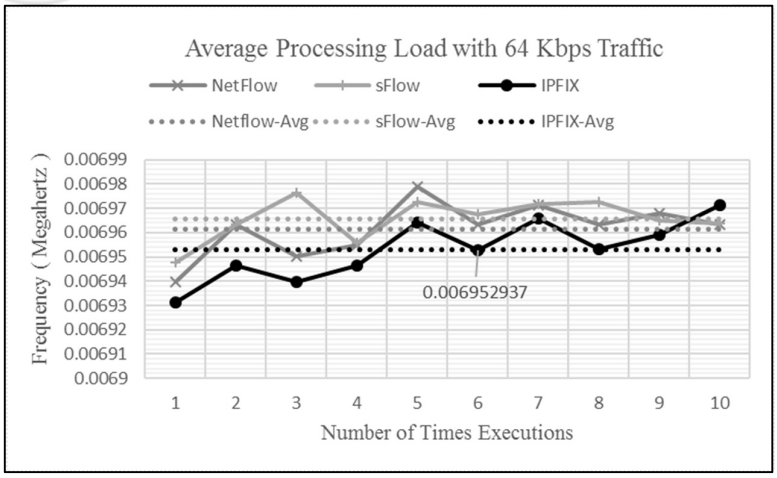

Figure 4.18. Average Processing load of different network probes with 64 Kbps traffic.

Table 4.3.

Average processing load collected in MHz during 1Mbps traffic transmission

No of test	NetFlow	sFlow	IPFIX
1	0.10810579	0.10830518	0.10801002
2	0.10845892	0.10855876	0.10850780
3	0.10840056	0.10837589	0.10797865
4	0.10830783	0.10853701	0.10825670
5	0.10874531	0.10836874	0.10836790
6	0.10855899	0.10860956	0.10833246
7	0.10865673	0.10895869	0.10868780
8	0.10855894	0.10867089	0.10873630
9	0.10883567	0.10861865	0.10859765
10	0.10855893	0.10876598	0.10845320
Avg. Load	**0.108518737**	**0.108576935**	**0.108392848**

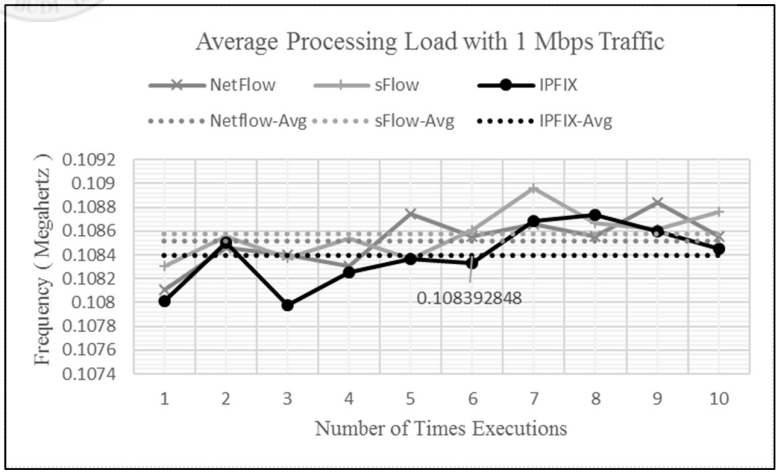

Figure 4.19. Average Processing load of different network probes with 1 Mbps traffic.

Following the benchmarking methodology results were obtained from number of tests performed. Figure 4.18 and 4.19 presents the average processing load of different flow technologies. It can be seen that IPFIX took slightly low average processing load. The average processing load of IPFIX is 0.006953 (Megahertz) MHz for 64 Kbps traffic and 0.10839 MHz for 1 Mbps traffic. The above simulation results were obtained on Intel 1.8GHz Quad core CPU based Linux machine running with network probe. In other words, average processing load is benchmark of our VXLAN based cloud network environment. We evaluated the CPU performance for our proposed VXLAN based cloud network environment to standardize the performance results in order to comparing the results with our proposed monitoring mechanism for evaluation and validation purpose.

4.7 Summary

This chapter presented design and development of VXLAN based cloud overlay network environment. Due to lack of resources, a hybrid technique has been used in this research project with combination of testbed and network simulation tools. The main advantages of using hybrid technique for evaluation of result in research include their capability for efficient modeling and simulation of large scale cloud computing data centers and federated clouds. The proposed cloud overlay network scenario consists of cloud controller, network controller, mininet emulator used on top of Open vSwitches on the hypervisors using OpenFlow to establish overlay networks. The proposed system was tested under limited resources using Linux machines for implementation and testing. Moreover, using benchmark methodology number of tests performed for analysis of flow based

technologies in order to evaluation and validation purpose of our proposed mechanisms in VXLAN based cloud overlay network environment. The VXLAN based packet capturing mechanism and filtering algorithm for cloud overlay network monitoring will be presented in Chapter Five.

CHAPTER FIVE
PACKET CAPTURING AND FILTERING MECHANISMS

The primary objective of this research project is to design a real time overlay network monitoring and performance analysis technique for large cloud infrastructure. The overall architecture of the proposed system consists of several stages, packet observation and selection, flow metering and export process, flow collection process and traffic analysis. This chapter presents the VXLAN based packet capturing and filtering mechanism for cloud overlay network monitoring as part of proposed system given in the Figure 5.1.

The organization of this chapter is as follows. Section 5.1 provides detail information of packet observation and selection with importance of selecting the right mode for packet capturing in large network environment. Section 5.2 discuss in detail packet capturing process. Section 5.3 explains each and every step taken on all packets for design of a VXLAN based filtering mechanisms, including VXLAN field match filtering mechanism (VFMFM) and VXLAN hash filtering mechanism (VHFM) described in detail. Section 5.4 presents experiment details and simulation results. Finally, Section 5.5 conclude the chapter by summarizing the main contribution of the chapter.

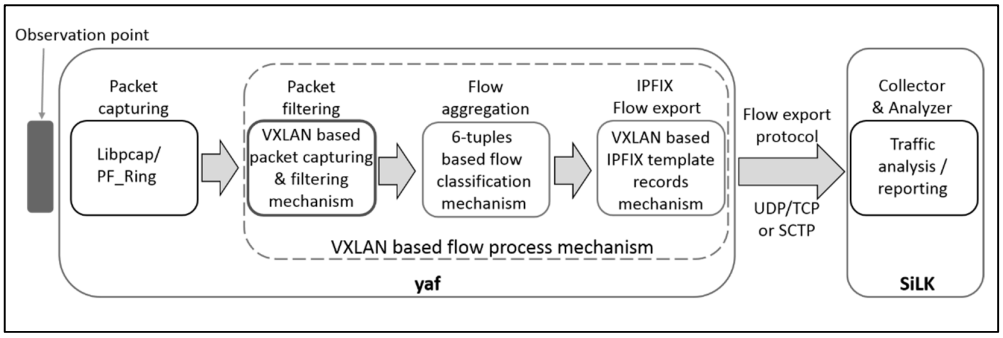

Figure 5.1. Enhanced IPFIX Flow Monitoring system (Packet capturing and filtering mechanism)

5.1 Packet Observation and Selection

The packet observation and selection stage consist of packet capturing, packet filtering and packet sampling in the proposed system. Note that all processing steps will be act on packets. The packet observation is first step to this architecture. Packet must be read on the line typically packet captured performed on Network Interface Card (NIC) which is carried out the packets. Before packets moved to the receiving host memory, there are several checks performed on the card buffer such as checksum errors to ensure that packet is received in original form. Due to high traffic output most of the packet capture performed on wired network, it can range from Local Area Network (LAN) to Wide Area Network (WAN) on the other hand, since the virtual environment rapidly growing in cloud environment, packet capturing of virtual networks become more common in cloud environment.

Packet capturing performed commonly in two modes in-line mode and mirroring mode. Selection of mode to capture the packets depends on bandwidth, type of analysis and monitoring environment. Therefore, it is a key to monitoring, to choose the packet capturing mode according to the requirements.

5.1.1 In-line mode

Network tap (test access point) nowadays mostly used in high speed networks for in-line mode traffic capturing. An external monitoring hardware device network tap placed between two network nodes for monitoring the link that mirror the packets of network traffic [138]. The capturing device directly connected between two nodes and make duplicate network traffic packets without any delay or loss in high speed networks. Furthermore, it forwards all mirror traffic to the dedicated capturing device for further processing. It eliminates the risk of drop packets and guarantee complete capture the packets even the 100% network link utilized. However, it cannot monitor intra-switch traffic and an additional cost required to purchase hardware network tap.

5.1.2 Mirroring mode

Now a day's most of the packet forwarding devices have ability to make copy or mirror the activity of network traffic that can be one or more ports through a Switch Port Analyzer (SPAN) port. The packet capture device connected to this SPAN port for collect the data. This is commonly referred to the port mirroring. It has ability to capture the intra-switch traffic and does not introduce the additional cost as network tap. Port mirroring performed well on low utilized networks. However, in high speed and heavily utilized network, port mirroring may introduce delay, packet drop and alter the content of the traffic stream [139]. It also creates burden on the CPU utilization of device while copy all data passing through ports.

In cloud environment virtual network rapidly gaining importance due to the widespread deployment of virtual machines (VM). In virtual environments things get a lot more complicated but deployment of packet capturing device is very similar to deployment in wired network. In cloud environment thousands of virtual machines interconnected to each other with virtual network and virtual network use virtual switches to interconnect the virtual machines [140]. Virtual switches working same as hardware switches which also support virtual network taps and port mirroring but are placed in virtual environment. Furthermore, in virtual environment traffic captured through in-line mode or mirroring mode thus mirrored traffic forwarded to physical ports and can be capture through dedicated packet capturing device outside the virtual environment.

5.2 Packet capturing process

Based on our lab scenario and above discussion we selected in-line mode for capturing the packets on high speed cloud network environment. It is necessary to mention here to build and operate reliable monitoring architecture required fully understand the performance of packet capturing process. There are many applications programming interface (APIs) and libraries are available, in open source Linux environment most reliable library libpcap [102] used for packet capturing. Since the operating system network stack performed general purpose networking therefore, libpacp library used for handover the packets from the NIC to the packet capturing application. The overall packet capturing process depends

on system performance as they added pre-packet processing overhead. To speedup this process several methods have been proposed [141]. One of the software based method PF_RING [142] proposed to bypass the network stack and avoid the pre-packet overhead to deal with higher packet rate. It improves the standard libpcap mechanism. It allows fast packet capturing mechanism without losing any packet loss and processing load. It is highly suitable for high speed networks.

5.3 Packet Filtering Mechanisms

Packet filtering is the technique which defines what kind of action perform on every single packet received from the observation point for the selection of particular packets. The role of packet filtering defined in RFC 5475 “separate all the packets having a certain property from those not having it” [143]. This step is adopted for selection of interested packets in our case VXLAN (Virtual eXtensible LANs) [144] packets. Typically, there are two types of filtering techniques for the selection of packets, one is property match filtering and second one is hash based filtering for packet selection [126]. In order to design the filtering mechanisms of cloud overlay packets. First, we need to understand the complete structure of VXLAN packet format which is defined in RFC 7348 [144].

.

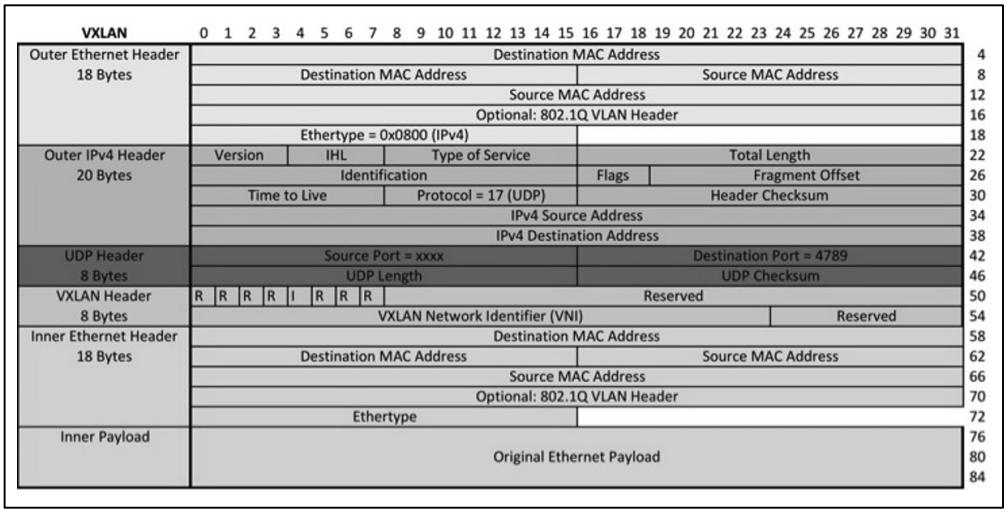

Figure 5.2. VXLAN Packet header detail [145].

Outer Ethernet Header: The outer Ethernet header is 18 bytes in length lets break these bytes further. First 6 bytes (48 bits) has a destination MAC address which represents the destination VTEP (VXLAN Terminal End Point). Second 6 bytes (48 bits) has a source MAC address of the source VTEP. Third 4 bytes (32 bits) are optional 802.1q VLAN Header. Bytes 17 and 18 (16 bits) represents the Ethernet type.

Outer IPv4 Header: The outer IPv4 header is 20 bytes in length and consist of byte 28 belongs to protocol UDP which represents the hex value equal to 0x11. Furthermore, bytes 31 to 34 (32 bits) source IP address of the source VTEP associated with the inner frame source and bytes 35 to 38 (32 bits) of outer destination IP address is the IP address of the destination VTEP.

Outer UDP Header: The outer UDP header is 8 bytes in length. Bytes 39 and 40 (16 bits) represents the source port which is dynamically assigned by the

originating VTEP. Bytes 41 and 42 (16 bits) represents the destination port is typically the well-known UDP port 4789.

VXLAN Header: The VXLAN header 8 bytes in length, lets break this header further which include following important fields. Note that we need to identify these fields for our proposed framework.

Flags: This is 8 bits field in length, where the 4th bit (I flag) is set to 1 for a valid VXLAN Network ID (VNI). The other 7 bits (R bits) are reserved fields and must be set to zero. The hex value is equal to 0x08.

VXLAN Network Identifier (VNI): This 24-bit value that provides a unique identifier for the individual VXLAN segment. The 24-bit VNI is used to identify Layer 2 segments and to maintain Layer 2 isolation between the segments. Hence, VMs in different VXLAN segments cannot communicate with each other.

Inner Ethernet Header: The original Ethernet frame header is 18 bytes and consists of the source and destination MAC addresses, an optional 802.1q header (VLAN ID) and Ethernet type. The frame is encapsulated using VXLAN, which adds the above additional headers and rest of bytes used for payload.

Since the VXLAN is new technology in cloud overlay network environment and packets are encapsulated over the network. Therefore, following two mechanisms for selection of VXLAN packets have been introduced. Figure 5.3 presents VXLAN based packet filtering mechanisms in order to selection of interested packets at observation point.

A. VXLAN Field Match Filtering Mechanism (VFMFM)

B. VXLAN based Hash Filtering Mechanism (VHFM)

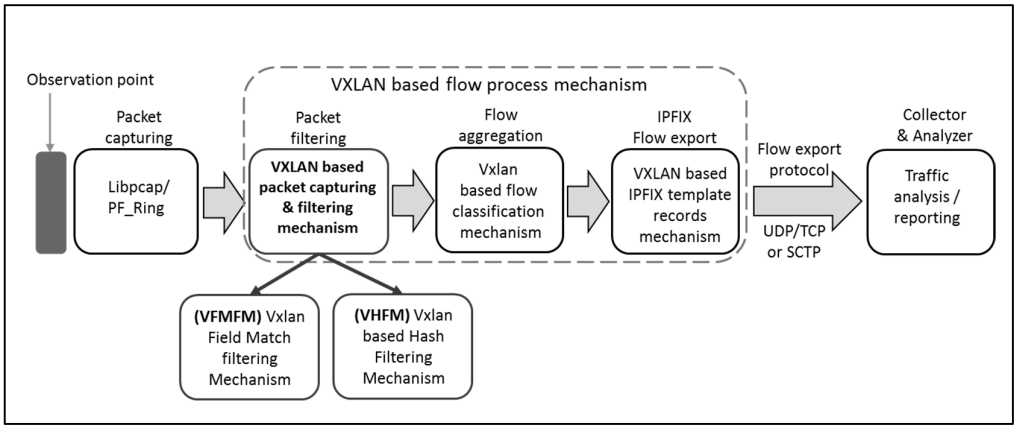

Figure 5.3. VXLAN based packet filtering mechanisms

5.3.1 VXLAN Field Match Filtering Mechanism (VFMFM)

Typically this type of packet filtering required property match filtering technique; a packet is selected if specific fields within packet are equal to a specified value or inside a specified value range [143]. Since the cloud overlay is a new technology and packets are encapsulated thus the most critical step is retrieving and sampling the VXLAN packets. Note that this is the key step of our research work to enhance the existing packet capturing system and add new technique to inspect every captured packet and select only VXLAN packets. All packets are read directly from the observation point with time stamped. Packets are inspected based on header instead of whole payload inspection to reduce the overhead and minimize

the load of packet selection stage. Thus, selected packet becomes an element of the output packet stream.

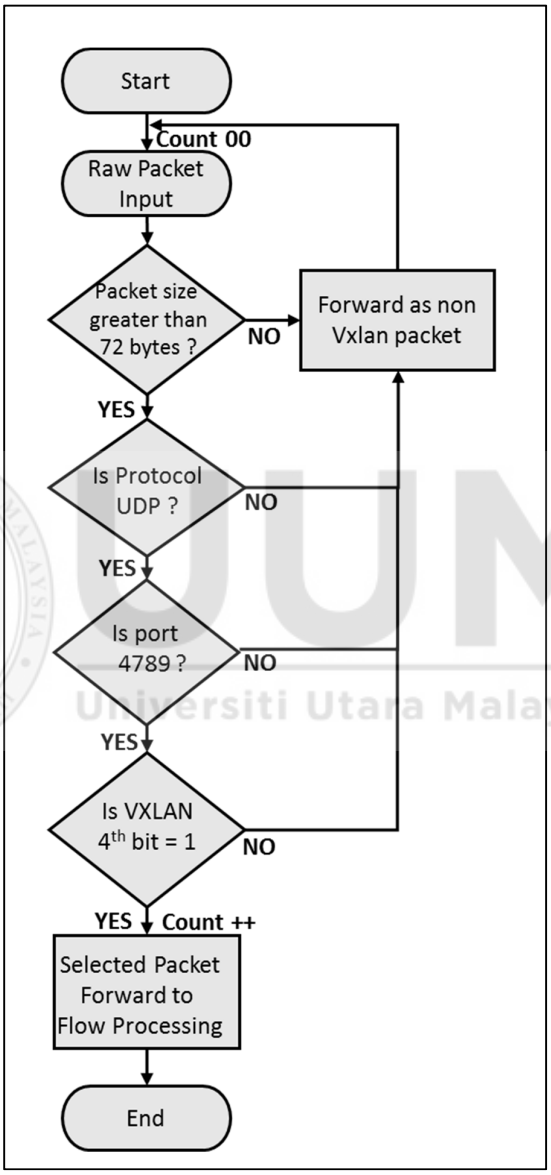

Figure 5.4. VXLAN based Field Match packet Filtering Mechanism

For VXLAN based packet filtering, following is detail description of the various mechanisms involved when a single packet arrives in to the line. To the selection of VXLAN packets, we must perform these procedures for each and every packet that arrives without dropping or altering even a single packet.

1. On first phase, each raw packet that arrives required packet size check, as minimum VXLAN packet size including all headers is 72 bytes without payload size. If the packet size less than 72 bytes then move to initial phase otherwise packet forward to next phase.
2. On second phase extract the Outer IPv4 header fields and check the packet protocol, as by default VXLAN packet use UDP for communication. If protocol is not UDP then move to initial phase otherwise packet forward to next phase.
3. On third phase extract the UDP header fields and check the destination port, by default VXLAN use 4789 destination port for communication. If destination protocol is not 4789 then move to initial phase otherwise packet forward to next phase.
4. On fourth phase open the VXLAN header and check the 4^{th} bit, as for valid VXLAN packet 4^{th} bit must be on out of first eight bits. If 4^{th} bit is not on then move to initial phase otherwise forward the packet for next phase.
5. Once all the checks successfully performed then select the packet from the input stream and packet count is incremented by one to account for the packet that just arrived. The selected packet will be forward to the flow processing.

The operation of proposed VXLAN based filtering mechanism has been functionally verified using simulation.

Suppose P indicating the packet flow of the network in which

$$P = (p_1, p_2, p_3, \ldots\ldots p_n)$$

the proposed mechanism executed using summation equation as shown in Equation 5.1 in order to extract only VXLAN packets from input packet stream.

$$P = \sum_{i=0}^{n} p_i \tag{5.1}$$

$$P = \{ p | p_i \in P \wedge p > 0 < i \leq n \tag{5.2}$$

where n is the number of filtered packets.

Examine the proposed VXLAN based packet filtering mechanism that follows to understand packet processing. When filtering is applied to check the received packets for VXLAN based cloud overlay packet match.

Suppose each packet Pi has header consist of K fields fk where k=1,2,3 .....K as shown in Figure 5.5. In Figure header field f* indicates the type of packet like VXLAN packet. Filtering mechanism checks this field, f*, to extract the overlay packet out of input packet stream.

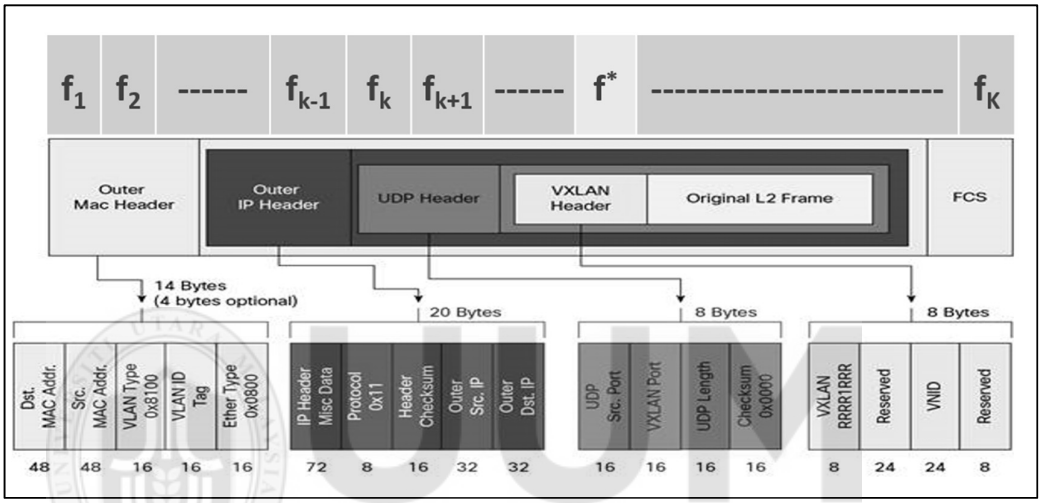

Figure 5.5. VXLAN Packet Header Fields [146].

An 8-byte VXLAN header that consists of a 24-bit Virtual Network Identifier (VNID) and some reserved bits as shown in Figure 5.4. The VXLAN header and original Ethernet frame together with in the UDP payload. The 24-bit VNID is used to identify Layer 2 segments and to maintain Layer 2 isolation between the segments. Filtering mechanism applied several steps to check the required fields, in order to extract the overlay packet out of input packet stream.

The pseudocode describes the implementation of this VXLAN Field Match Filtering Mechanism (VFMFM). The pseudocode of VFMFM based packet selection is given in Algorithm 5.1.

Algorithm 5.1 VFMFM based packet selection algorithm

```
OPcount is initialized to zero
Arrival of new packet Pi
Open the new packet  Pi and check the packet size
If (the packet size < 72 bytes) then       /* Minimum VXLAN packet size is 72
                                                                    bytes */
 No action

  Else If (the packet size > 72 bytes) then
  Lookup the protocol
  If (the packet protocol != UDP) then      /* Standard VXLAN packet is UDP */
     No action

          Else If (the packet protocol = UDP) then
          Lookup the destination port
          If (destination port != 4789) then
          No action
```

```
Else If (destination port = 4789) then
Open the VXLAN header of the packet Pi
Lookup the 4th bit
If (4th bit is != 1) then
No action

        Else If (4th bit is = 1) then
        Select the packet and
        Retrieve the 24-bit value of VNI        /* VNI value
                        started from 51 bytes to 53 bytes... */
        Increment OPcount
End If
```


5.3.2 VXLAN based Hash Filtering Mechanism (VHFM)

A Common hash function consist of hash domain and selection range. The hash domain comprises part of the packet content that is invariant fields within the packet header. Typically, this type of packet filtering required invariant fields with in the packet header; a packet is selected if specific fields within packet are equal to a specified invariant selection range. However, for security reason ports of communication might be changed in cloud overlay network environment, in that case VFMFM filtering will not able to capture the VXLAN based traffic because filtering attributes are changed. Therefore, only hash function can identify the invariant fields with in the packet for selection of VXLAN packet in the network

flow. All packets are read directly from the observation point and inspected based on given hash selection range. Thus, selected packet becomes an element of the output packet stream.

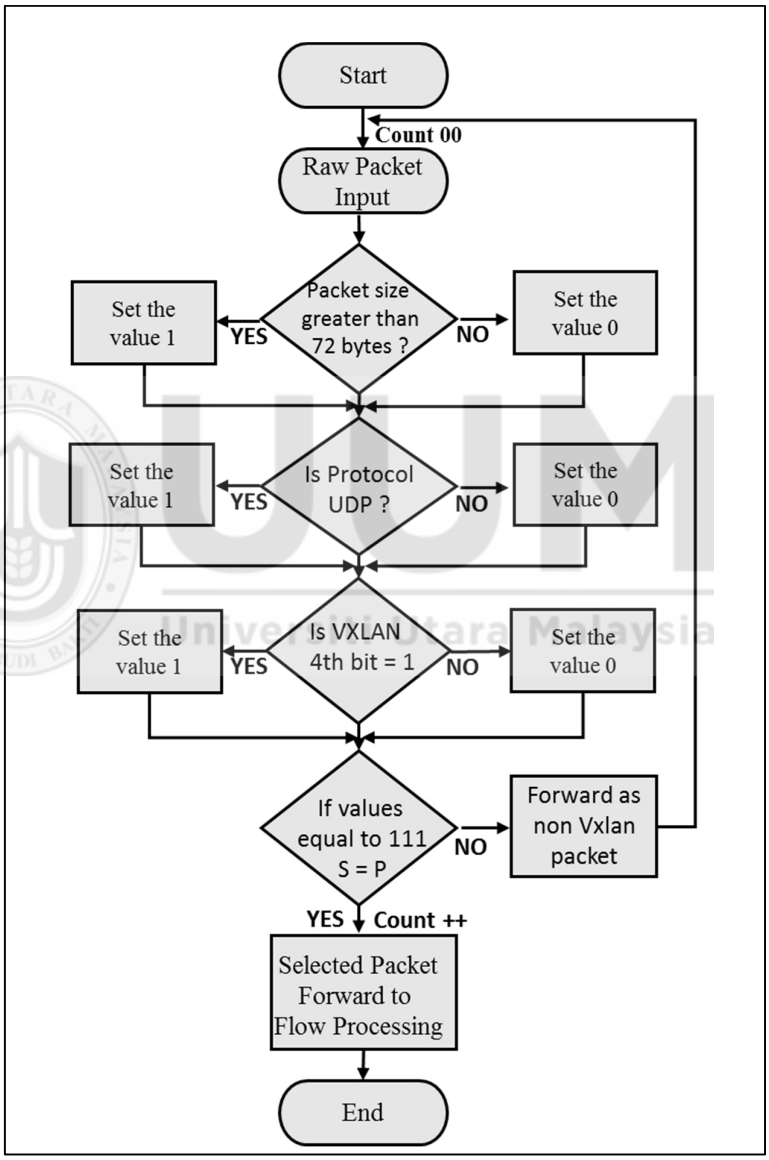

Figure 5.6. VXLAN based Hash Filtering Mechanism for packet selection

Following hash function executed on each packet that arrived on the observation point. The given hash function derived from [147], where author used hash functionality for post sampling analysis of distinct trajectories once the samples are reported on every hop on the network path. Whereas, we used hash functionality to compare the values in the packet header for VXLAN based packet selection.

Hash Function $h(\phi(x))$

Where $\phi(x)$ is an invariance function of the packet header contents i.e. fields (bits) of header that are not modified upon routing (forwarding /during transmission). Thus $\phi(x)$ in the range of hash function.

The selection set 'S' in comprises of followings three fields

1. In byte 21 and 22 Header length >72 bytes
2. 28th byte equals to protocol UDP value in hex 0x11
3. In 47th byte, 4th bit is on, 00001000 and equals to value in hex 0x08

	21&22nd bytes	28th byte	47th byte
P =	Header Size > 72 bytes	0x11	0x08

$$h(\phi(x)) = \begin{cases} 1, & h(\phi(x)) \in S = P \\ 0, & otherwise \end{cases} \quad (5.3)$$

The hash function used for packet selection only based on header fields. The proposed mechanism is used for comparing pattern 'P' only, hence hash functionality is shorter and faster as compared to given hash function [147] for packet selection.

The pseudocode describes the implementation of this VXLAN based Hash Filtering Mechanism (VHFM). The pseudocode of VHFM based packet selection is given in Algorithm 5.2.

Algorithm 5.2 VHFM based packet selection algorithm

```
OPcount is initialized to zero
Arrival of new packet Pi
Open the new packet  Pi and check the packet size

If (the packet size > 72 bytes) then        /* Minimum VXLAN packet size is 72
bytes */
   set the value = 1
   Else set the value = 0
      Else If (the packet protocol = UDP) then
         set the value = 1
            Else set the value = 0
                Else If (4th bit is = 1) then
                set the value = 1
                Else set the value = 0
                    If S = P then        */ where S is equal to value 111*/
```

Select the packet and

Retrieve the 24-bit value of VNI

Increment OPcount

Else

no action

End If

5.4 Experimental Results

To illustrate the difference between standard IPFIX flow monitoring and proposed VXLAN based flow monitoring for overlay cloud networks, simulation performed to evaluate the performance of VFMFM and VHFM filtering mechanisms, refer to the picture 4.1 configuration. Traffic generated between different network segments using benchmarking methodology with transmission duration 60 minutes using the dataset presented in Table 4.1. Traffic generated through well-known network tool iperf [98] between different virtual machines for filtering mechanisms performance measurement in different network segments. A plugin developed for both filtering mechanisms as a part of this project and executed with open source tool YAF [90]. YAF has ability to capture live traffic from an interface using pcap into bidirectional flows, then forward these captured packets to further processing for flow aggregation and export process.

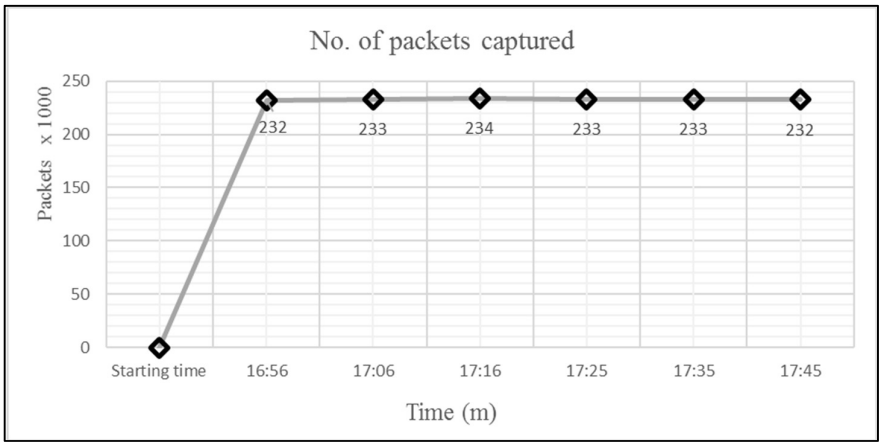

Figure 5.7. Standard IPFIX Monitoring with Packet Capture Detail (64 Kbps)

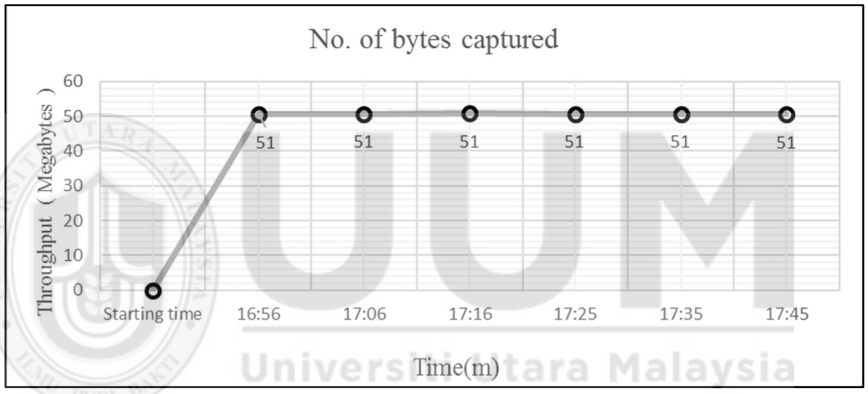

Figure 5.8. Standard IPFIX Monitoring with Bandwidth Detail (64 Kbps)

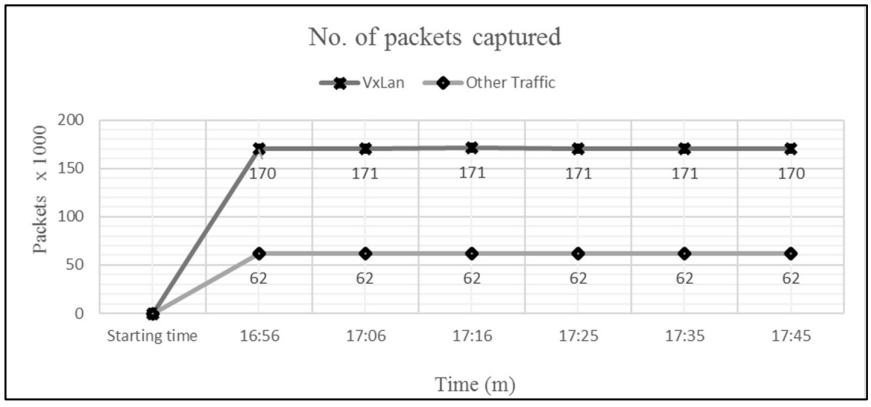

Figure 5.9. VXLAN based Filtering Mechanism with Packet Capture Detail (64 Kbps)

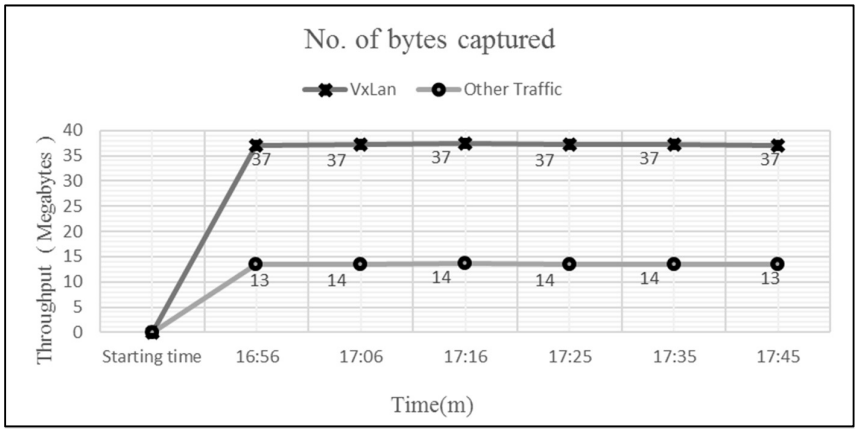

Figure 5.10. VXLAN based Filtering Mechanism with Bandwidth Detail (64 Kbps)

The experiment results of the standard IPFIX monitoring tool are shown in Figure 5.7 and 5.8. The standard IPFIX monitoring tool only captures the total number of packets and bandwidth but could not identify the VXLAN based tunnel traffic in virtual cloud network environment. On the other hand, Figure 5.9 and 5.10 represents the result of proposed VXLAN based filtering monitoring mechanism, it can be seen that proposed filtering mechanisms are able to captured VXLAN packets and differentiate the VXLAN traffic and other traffic. The above results are based on following the benchmark methodology with 64 Kbps traffic of dataset. Likewise, another simulation performed with the same environment to evaluate the performance with 1 Mbps dataset as given in Table 4.1.

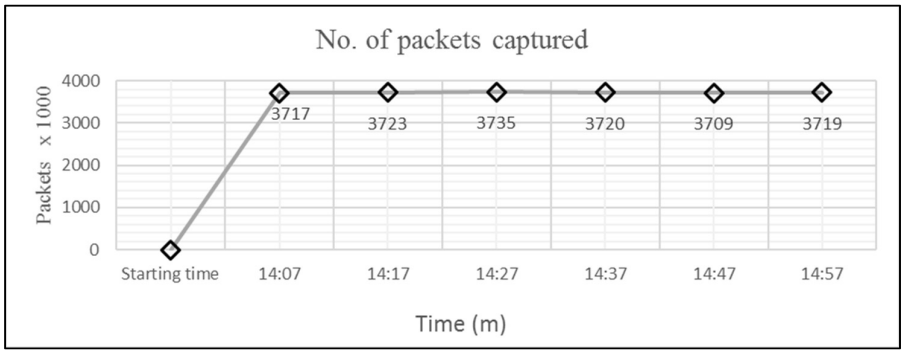

Figure 5.11. Standard IPFIX Monitoring with Packet Capture Detail (1 Mbps)

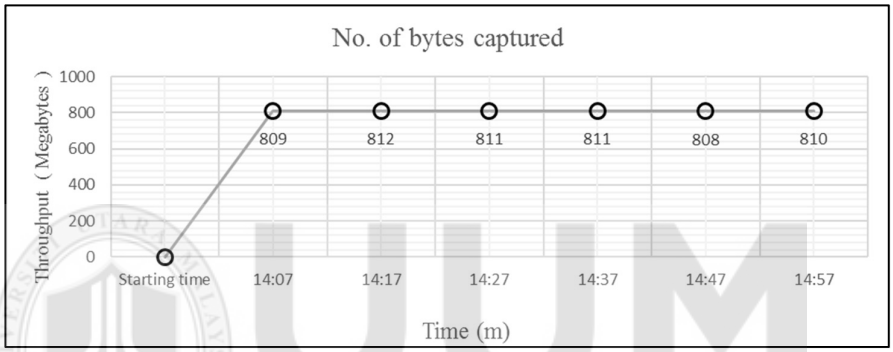

Figure 5.12. Standard IPFIX Monitoring with Bandwidth Detail (1 Mbps)

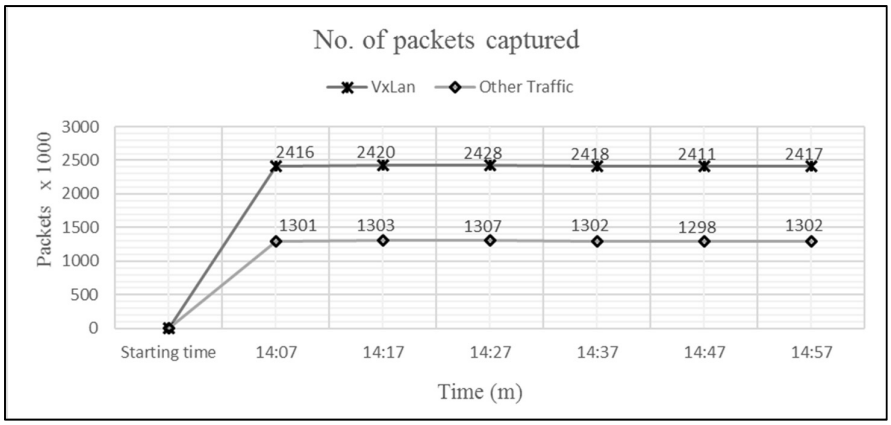

Figure 5.13. VXLAN based Filtering Mechanism with Packet Capture Detail (1 Mbps)

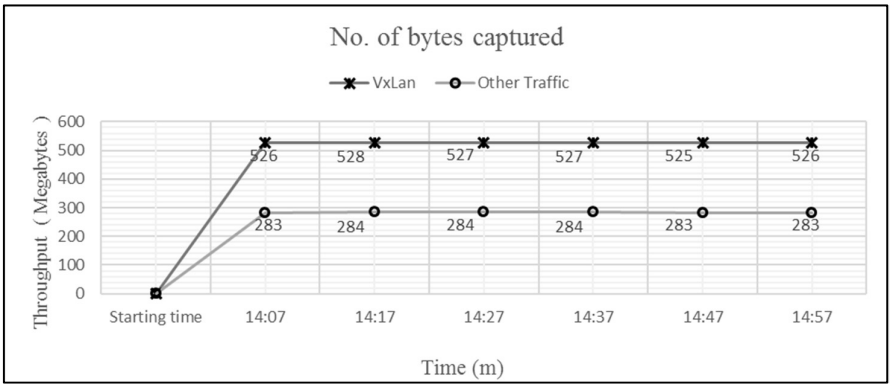

Figure 5.14. VXLAN based Filtering Mechanism with Bandwidth Detail (1 Mbps)

Figure 5.11 and 5.12 shows Standard IPFIX monitoring traffic results and Figure 5.13 and 5.14 represents proposed VXLAN based filtering mechanism traffic. In the results, VXLAN traffic represents total number of packets/bytes captured through proposed packet capturing and filtering mechanism for overlay networks with time stamp duration, and remaining traffic shown as other traffic, which is non VXLAN traffic. However, standard IPFIX monitoring tool not able to capture the virtual tunnel traffic and only shows as total traffic. Hence, our measurements have confirmed that proposed mechanism able to capture the live tunnel traffic and also clearly identify VXLAN packets and distinguish the other traffic in high speed cloud virtual network environment.

Besides that, experiment results CPU processing load also been evaluated for our proposed VFMFM and VHFM filtering mechanisms. The CPU processing load data were collected with same simulation environment, in order to compare the results with standard IPFIX processing load as mention in section 4.6 in Chapter four.

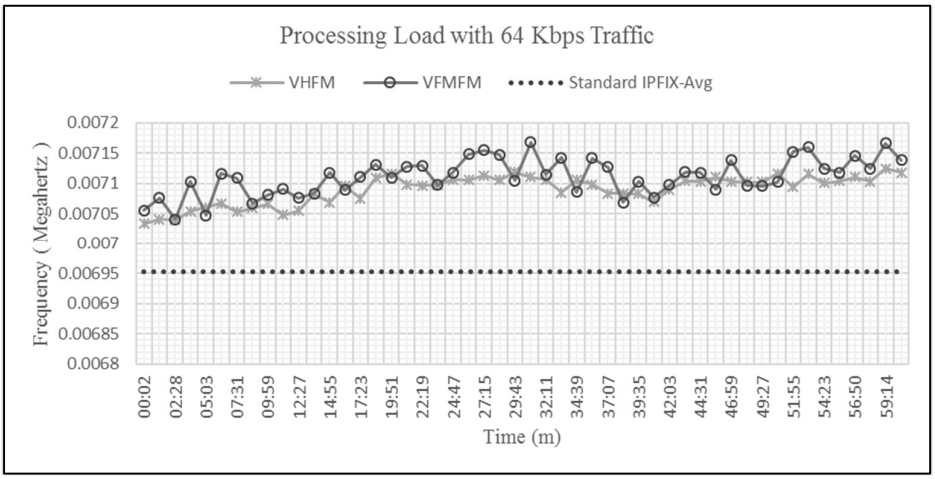

Figure 5.15. Processing load of VHFM and VFMFM Filtering Mechanism (64 Kbps)

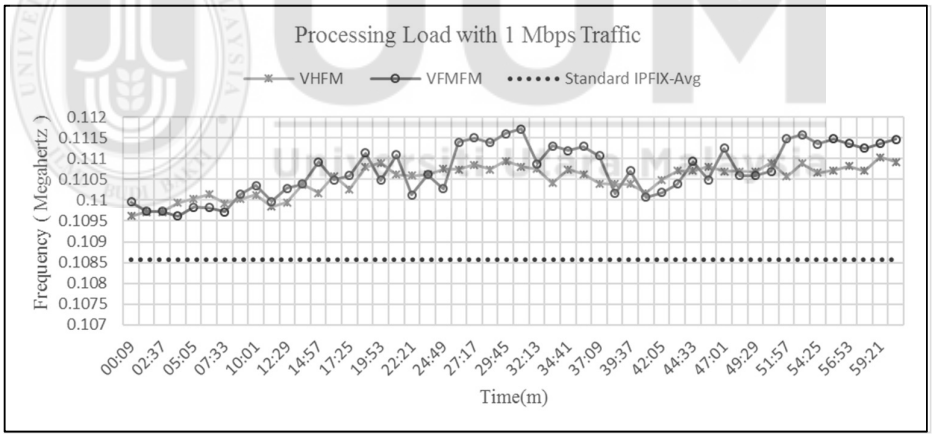

Figure 5.16. Processing load of VHFM and VFMFM Filtering Mechanism (1 Mbps)

Table 5.1 presents 64 Kbps traffic processing load data, collected during experiment performed with proposed filtering mechanisms VFMFM and VHFM.

Table 5.1.

Processing load with filtering mechanisms collected in MHz during 64 Kbps traffic transmission.

No of test	VFMFM	VHFM
1	0.00711123	0.007090142
2	0.00721079	0.007161043
3	0.00716812	0.007132683
4	0.00725346	0.007146863
5	0.00723212	0.007168134
6	0.00725346	0.007153953
7	0.00722501	0.007168134
8	0.00722501	0.007161043
9	0.00718946	0.007139773
10	0.00723923	0.007146863
Avg. Load	**0.00721079**	**0.007146863**

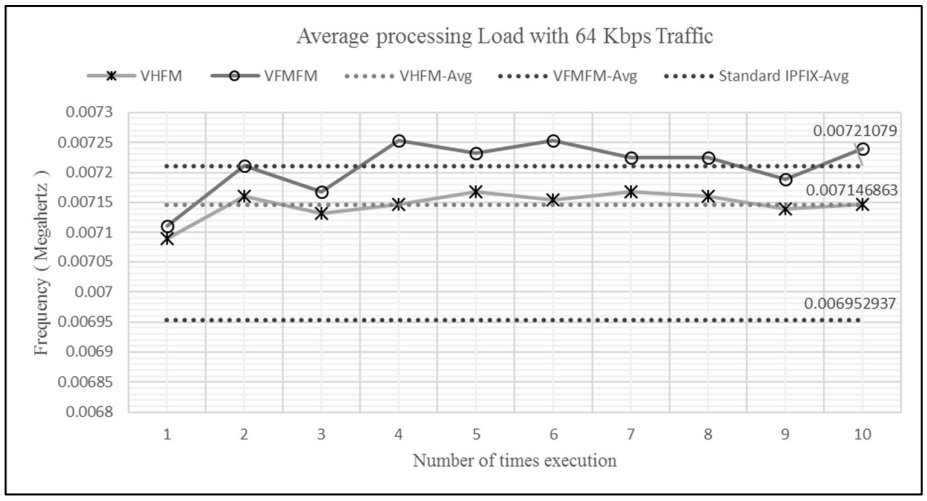

Figure 5.17. Average Processing load analysis of VHFM and VFMFM (64 Kbps)

Figures 517 and 5.18 shows average processing load of both VHFM and VFMFM filtering mechanisms with 64 Kbps traffic load.

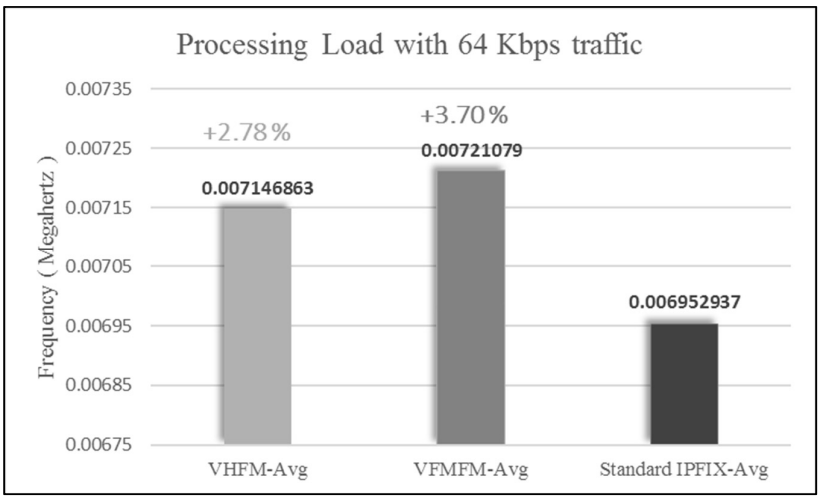

Figure 5.18. Comparison of VHFM and VFMFM mechanisms processing load with 64 Kbps traffic.

It can be seen that both filtering mechanisms took more processing load as compare to average IPFIX processing load. While to ensure the results are accurate, same simulation with same environment repeated 10 times to calculate the average processing load of both filtering mechanisms presented in Table 5.1 for 64 Kbps traffic load. Figure 5.18 shows average processing load of both VHFM and VFMFM mechanisms on 64 Kbps traffic. It can be seen that VFMFM took on average 3.70% percent more processing load as compare to average IPFIX processing load, while VHFM took 2.78% percent more load as compare to IPFIX, however, VHFM took slightly low processing load, it is because of hash functionality.

Table 5.2 presents 1 Mbps traffic processing load data, collected during experiment performed with proposed filtering mechanisms VFMFM and VHFM.

Table 5.2.

Processing load with filtering mechanisms collected in MHz during 1 Mbps traffic.

No of test	VFMFM	VHFM
1	0.11072301	0.11049572
2	0.11282674	0.111600677
3	0.11160879	0.111048198
4	0.11293747	0.111379685
5	0.11282674	0.111932164
6	0.11293747	0.111490181
7	0.11326964	0.112153155
8	0.11338036	0.112153155
9	0.11282674	0.111490181
10	0.11271602	0.111379685
Avg. Load	**0.112605298**	**0.11151228**

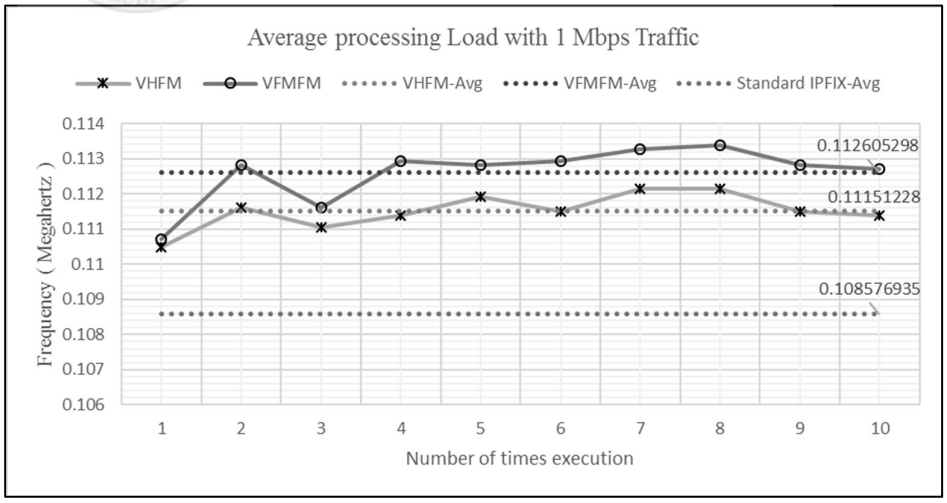

Figure 5.19. Average Processing load analysis of VHFM and VFMFM (1 Mbps)

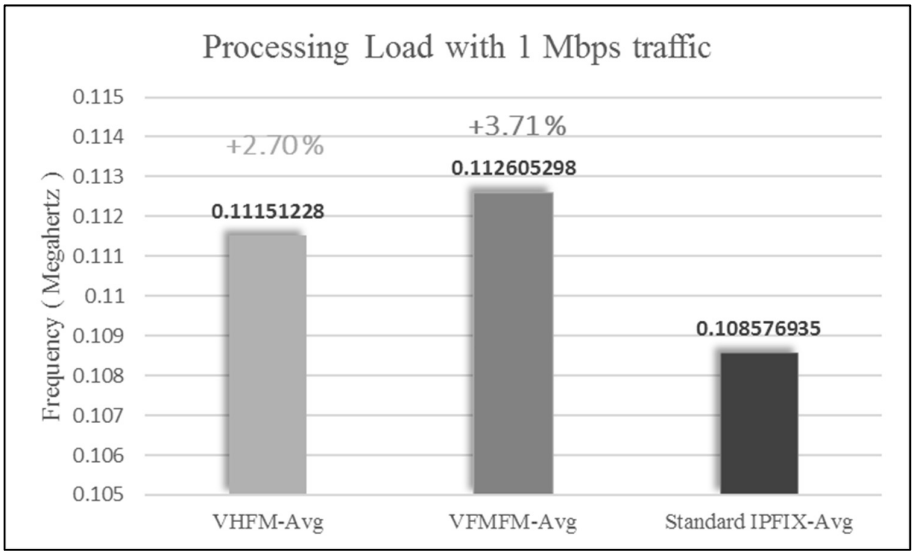

Figure 5.20. Comparison of VHFM and VFMFM mechanisms processing load with 1 Mbps traffic

Figure 5.19 and 5.20 presents VHFM and VFMFM mechanisms processing load with 60 minutes simulation on 1 Mbps traffic load. The average processing load of both filtering mechanisms presented in Table 5.2 for 1 Mbps traffic load. It can be seen that VFMFM took 3.71% percent more CPU utilization as compare to average IPFIX CPU utilization. Whereas VHFM took 2.70% percent more as compare to standard IPFIX but slightly low CPU utilization as compare to VFMFM. We have been evaluated both filtering mechanisms and presented results are evidence of that. However, VHFM performance faster than VFMFM and these results are justified because filtering mechanisms took more processing load as compare to standard IPFIX monitoring.

5.5 Summary

This chapter presented VXLAN based packet capturing and filtering mechanism for cloud overlay network monitoring system as part of this research project. The complete process from packet observation to selection has been discussed in detail. The VXLAN based both filtering mechanisms VHFM and VFMFM were tested and the simulation results shows. In the results, VXLAN traffic represents total number of packets/bytes captured through proposed packet capturing and filtering mechanism for overlay networks with time stamp duration, and remaining traffic shown as other traffic, which is non VXLAN traffic. However, standard IPFIX monitoring tool not able to capture the virtual tunnel traffic and only shows as total traffic. Thus, our measurements have confirmed that proposed mechanism able to capture the live tunnel traffic. Furthermore, proposed mechanism clearly identifies VXLAN packets and distinguish the other traffic in high speed cloud virtual network environment. Hence, it is confirmed that proposed mechanisms functionality working as expected. Proposed filtering mechanisms can provide network operators with detailed information about the traffic traversing a link. Besides that, experiment results CPU processing load also been evaluated for our proposed VFMFM and VHFM filtering mechanisms. The flow classification algorithms and VXLAN based data records for IPFIX messages template will be presented in Chapter Six.

CHAPTER SIX
ENHANCED IPFIX FLOW PROCESSING MECHANISM

Chapter Five presented packet capturing and filtering mechanism for the proposed monitoring system which were working as expected and next stage is to develop flow classification and IPFIX template record mechanism for this research project. This chapter introduces flow classification mechanisms and IPFIX template record mechanisms for VXLAN based overlay network, as a part of proposed system given in the Figure 6.1. The main goal of this research project to develop a monitoring system with enhanced IPFIX flow processing mechanisms for large and high speed cloud overlay networks. The organization of this chapter is as follows. Section 6.1, 6.2 and 6.3, provided principles of flow processing stage and introduced the main contribution of this chapter, the VXLAN based flow processing mechanism including 6-tuple based flow classification mechanism and adaptable flow classification mechanism (AFCM) for enhanced IPFIX flow processing mechanism. Section 6.4 explains in detail flow cache management and flow entries expiration. Section 6.5 explains in detail IPFIX messages mechanism. Section 6.6 and 6.7 introduced the second main contribution of this chapter the VXLAN and AFCM based flow data record template mechanisms for IPFIX messages. Section 6.8 and 6.9 provided in detail IPFIX message export process and flow collection process. Section 6.10 provides overlay traffic performance analysis with detail simulation environment and experiment results. Finally, Section 6.11 concludes the chapter providing a brief summary of the complete chapter.

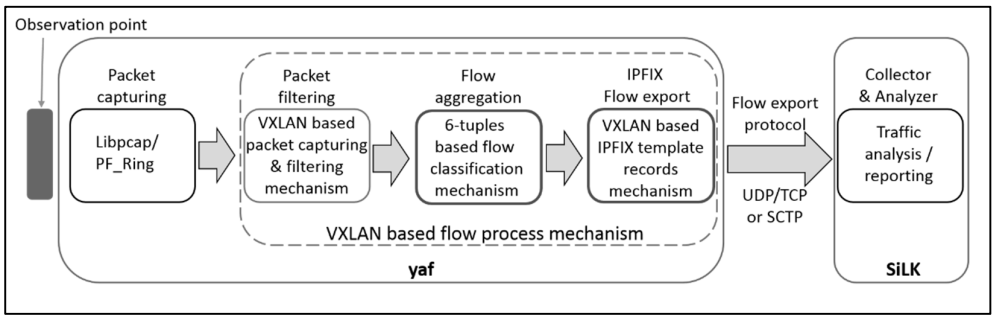

Figure 6.1. Enhanced IPFIX Flow Monitoring system For VXLAN Based Cloud Overlay Networks (Flow Classification and Message Template Mechanisms)

6.1 Flow Processing

The flow processing stage consist of packet aggregation, flow cache, flow selection and transport protocol. After the selection of filtered packets, cloud overlay packets will be aggregated into network traffic flows for temporary stored in flow cache. Network traffic flow describe as a sequence of packets between two endpoints based on key filed. Typically flow pattern based on 5-tuple which represents the set of five different key values, describes in Figure 6.2. It includes a source IP address and source port number, destination IP address and destination port number and the protocol in use [148]. If key fields are matched in the new captured packet which belongs to existing flow, packet will be added to existing flow and information is updated accordingly. On the other hand, if key fields are not matched any of existing flow, new flow will be generated and stored in the flow cache. Flow generation and update steps can perform repeatedly for flow aggregations.

Typical Flow pattern based on 5-tuple + Timing and Data statistics

Typical Flow pattern based on 5-tuple

Source IP	Source Port	Destination IP	Destination Port	Protocol
172.16.10.10	54351	172.16.20.20	4789	UDP

Timing and Data stats

Start Time	End Time	Packets	Bytes
2017/06/22T00:04:58.756	2017/06/22T00:05:18.325	54	134

Figure 6.2. Typical Flow pattern based on 5-tuple + Timing and Data Statistics

However, typical 5-tuple based flow pattern cannot fulfil our requirements in case of VXLAN based flow generation. Therefore, following two new mechanisms introduced in order to enhanced flow aggregation process in IPFIX for cloud overlay network monitoring. Figure 6.3 presents newly introduced VXLAN based flow classification mechanisms.

A. 6-tuple based Flow Pattern and Classification Mechanism

B. Adaptable Flow Classification Mechanism (AFCM)

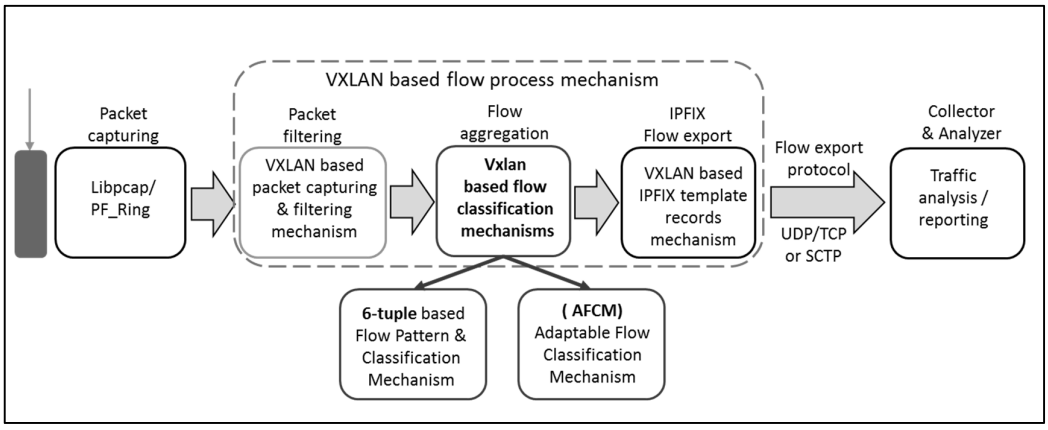

Figure 6.3. VXLAN based Flow Classification Mechanisms

6.2 VXLAN based 6-tuple Flow Pattern

Since the research focus is to design a mechanism for cloud overlay network monitoring for high speed network. The typical 5-tuple base flow pattern not fulfil the requirement of our need. The VXLAN based flow classification required more fields instead of 5-tuple fields. Since the VXLAN based cloud overlay network traffic uses network identifier (VNI) unique value for communication between two tenants, where each tenant creates their own dynamic overlay network for communication of other tenant. The VNI field has 24 bits and can identify a maximum of 16M VXLAN segments. The key to monitoring VXLAN based overlay network traffic is to identify the VNI value. Therefore, we introduced new flow pattern called VXLAN based 6-tuple flow, which represents set of six key field values. We added new VNI key field on traditional flow key pattern which makes its unique 6-tuple VXLAN based flow pattern. The fields include a source IP address and source port number, destination IP address and destination port number, protocol and new added field VNI. If any of the field changed a new flow will be generated. Figure 6.4 represents the new VXLAN based 6-tuple flow pattern.

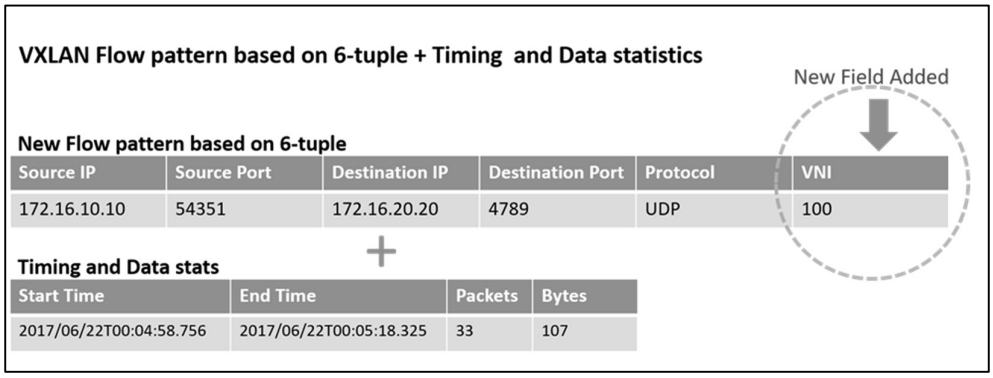

Figure 6.4. VXLAN Flow pattern based on 6-tuple + Timing and Data Statistics

Furthermore, each separate flow have an entry associated with non-key fields including flow start time, end time, total number of packets and total bytes. All active network traffic flows information maintained in flow cache.

6.2.1 VXLAN based 6-tuple Flow Classification

Flow classification is used to map each input packet to the flow it belongs to. This operation is necessary as the processing of each input packet is usually done at VXLAN based packet filtering mechanism. After the filtering each packet that arrives in the flow classifier the relevant 6-tuples header fields are extracted. The 6-tuples header field information in the arriving packet is compared with the existing flow entries if there is no match entry found. New flow will be generated based on 6-tuples VXLAN pattern. In the event of existing entry match, the existing flow entry is updated with information from this newly arrived packet and several fields are also updated. The packet count is incremented by one to account for the packet that just arrived. The byte count is incremented by the number of bytes of data present in the packet. The timestamp is updated with the current time,

to indicate that a new packet just arrived for this flow. The timestamp is used to age out old flow entries. The flow classification steps describe in Figure 6.5.

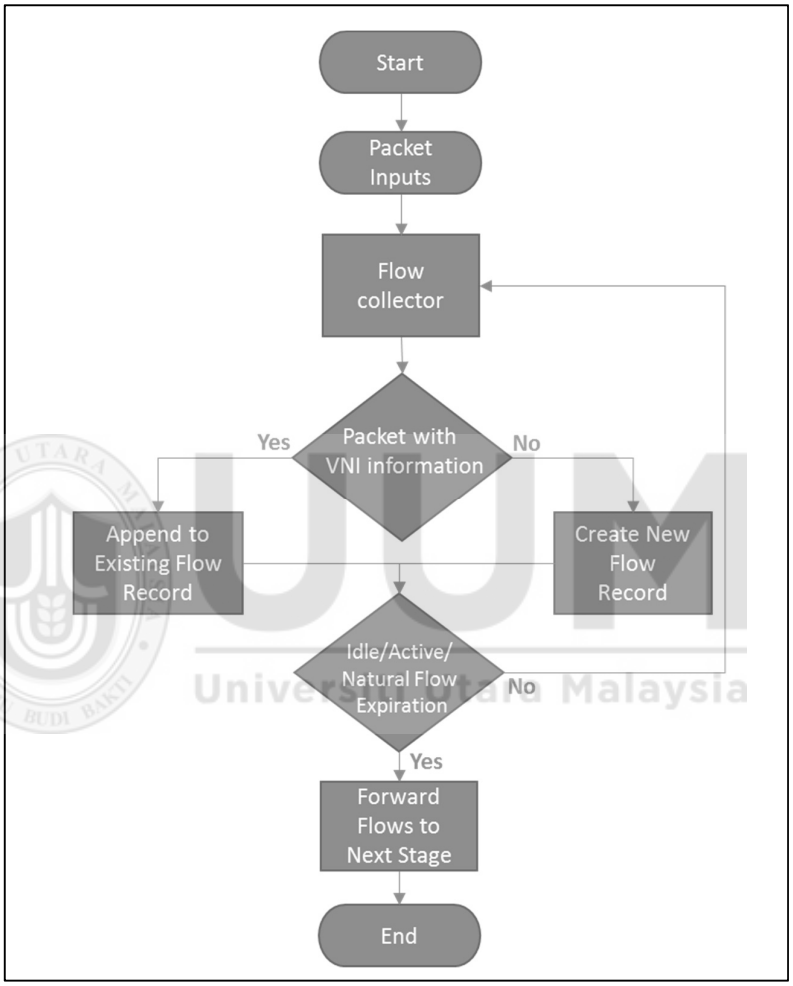

Figure 6.5. VXLAN based Flow classifier

The pseudocode describes the implementation of flow classification mechanism based on 6-tuples pattern is given in Algorithm 6.1.

Algorithm 6.1 VXLAN based flow pattern algorithm

```
OPcount is initialized to zero
Call the flow flush timer and initialized to zero
OPScount is initialized to zero     /* Overlay Packet size*/
Arrival of new packet vPi
Open the new packet  vPi and lookup the 6-tuple key fields
If (6-tuple key fields found) then
check the VXLAN based flow-group-pattern
    if ( No VNI based flow-group-pattern found) then
    Make new flow-group-pattern [Source, Destination IP, PortIn,
    PortOut, VNI]
    Else
        if( Group-pattern is seen) then
        Send/Add packet to existing Active flow-group
            Increment OPcount
            Increment OPScount
        Else
            Check the flow flush timer
        If ( flush timer expire) then
              Make new flow-group-pattern
End if
```

Furthermore, 6-tuple based flow classification mechanism takes slightly more processing load on flow aggregation process in IPFIX. It is due to encapsulation layer involve in order to extract the packet detail and that is fairly justified to

generate the 6-tuple based flow aggregation mechanism for detailed cloud overlay network monitoring. However, in order to further enhanced flow classification mechanism for cloud overlay network monitoring an adaptable flow classification mechanism (AFCM) introduced.

6.3 Adaptable Flow Classification Mechanism (AFCM)

Flows defined as a set of packets that are observed at particular observation point in the network that belongs to either typical 5-tuples or VXLAN based 6-tuples or any other combination of tuples depends on required application monitoring data. Every distinctive flow based on tuples information stored in the cache for certain time, later these large amounts of data exported for further analysis. Typical flow based monitoring probes often expert 5-tuples key and many non-key fields flow records, results in a large amount of high traffic volume. As the network is growing day by day in cloud data centers, a small or medium size enterprise level organization monitoring probes easily generate gigabytes of flow records on every day. Hence, the reduction of monitoring data required in order to export the flows on the network to the collector.

Nowadays flow aggregation mechanism mostly deployed at collector end. After collecting the monitoring data at collector from exported flows, data is analyzed according to the application needs and unnecessary information is discarded from monitoring data. However, flow aggregation mechanism at collector end does not reduce the data generated by monitoring probes. In order to limit the amount of exported monitoring data we introduced an Adaptable Flow Classification

Mechanism (AFCM) by discarding unnecessary fields from the flow information and retain only key fields as per monitoring application requirement before the flows are exported. As a result, AFCM flow information portray the same results of monitoring data which is required as per application needs with less amount of data transmission and bandwidth consumption and help to minimize the load on flow processing stage.

In order to achieved this, we classify the flow pattern which defined flow key fields and non-key fields and specifies which key fields retain and which fields are discarded using VXLAN based AFCM rule. An AFCM rule defines how to deal with every incoming packet field in the flow classifier to aggregate the flows. An Adoptable Flow Classification Mechanism only applicable on those incoming packets who have all specified fields which are mention in AFCM flow rule. The required flow key fields are directly derived from the packet by retaining and discarding fields according to AFCM flow rule. The selected AFCM based flows forwarded for further process. Table 6.1 presents the VXLAN based Adoptable Flow Classification Mechanism rule with discarded and retains key fields. It is necessary to mention here Adoptable Flow Classification Mechanism have flexibility to adopt any flow pattern according to the application requirement defining by as a new rule.

Table 6.1.

VXLAN based AFCM key and non-key fields

ID	Fields name	bytes	Key and non-Key fields	AFCM
1	octetDeltaCount	8	Non-key	
2	packetDeltaCount	8	Non-key	
4	protocolIdentifier	1		Discard
7	sourceTransportPort	2		Discard
8	sourceIPv4Address	4	Key	Retained
11	destinationTransportPort	2		Discard
12	destinationIPv4Address	4	Key	Retained
152	flowStartMilliseconds	8	Non-key	
153	flowEndMilliseconds	8	Non-key	
1001	VNILable	3	Key	Retained

6.3.1 VXLAN based Adaptable Flow Classification Mechanism

Many applications not required granularity of 5-tuple for the processing of monitoring data. For instance cloud provider and customer required to calculate the billing based on the network traffic usage. In that case monitoring of total traffic volume sent and received has to be determined based on customer host or subnet of hosts. In other words, we only required source and destination IP key fields in order to calculate the traffic usage. While VXLAN based cloud overlay network monitoring required source machine IP address and destination machine IP address including VNI information to determine that when tunnel was created in order to communicate with another machine. Figure 6.6 presents the VXLAN based Adaptable Flow Classification Mechanism flow pattern. Where Source IP,

Destination IP and VNI key fields are retained and extracted from packet and remaining Destination port, Source port and protocol fields are discarded and not derived from packet. Moreover, each separate flow has an entry associated with non-key fields including flow start time, end time, total number of packets and total bytes are updated accordingly.

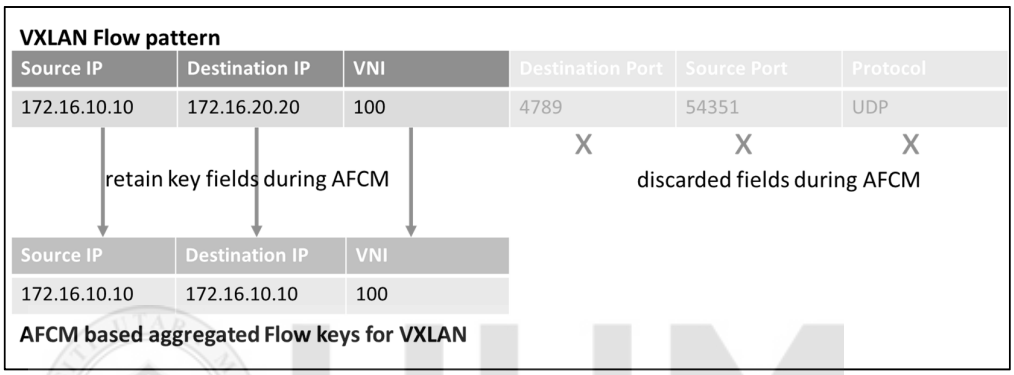

Figure 6.6. Adaptable Flow Classification Mechanism Flow pattern

AFCM has ability to reduce the amount of monitoring data before exporting to collector for data analysis with same results as required by monitoring application. Furthermore, AFCM process help to reduce the CPU utilization as compare to standard IPFIX process.

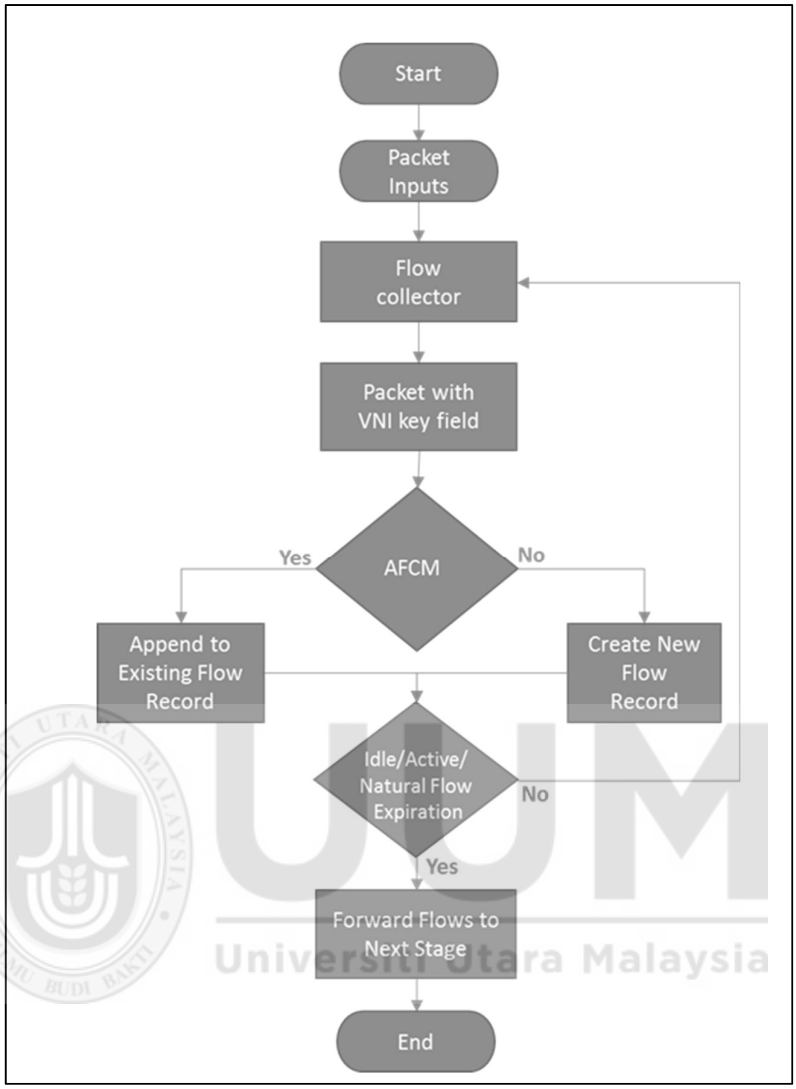

Figure 6.7. AFCM based VXLAN Flow classifier

The pseudocode describes the implementation of Adoptable Flow Classification Mechanism based VXLAN pattern is given in Algorithm 6.2.

Algorithm 6.2 VXLAN based AFCM flow pattern algorithm

```
OPcount is initialized to zero
Call the flow flush timer and initialized to zero
OPScount is initialized to zero     /* Overlay Packet size*/
Arrival of new packet vPi
Apply the VXLAN based AFCM rule
Check the AFCM key fields value of the packet vPi
   if the VXLAN based AFCM key fields not found in the packet header
   then
   no action
       Else
       if the VXLAN based AFCM key fields matched then
       Select only key fields as per AFCM rule
   [Source IP, Destination IP, VNI]
   and discard the remaining fields
       Select the packet and forward to flow aggregation process and
       check the AFCM based flow-group-pattern
       if ( No AFCM based flow-group-pattern found) then
            Make new flow-group-pattern [Source IP, Destination IP, VNI]
            Else
            if( Group-pattern is seen) then
            Send/Add packet to existing Active flow-group
                 Increment OPcount
                 Increment OPScount
                 Else
                 Check the flow flush timer
```

If (flush timer expire) then

Make new flow-group-pattern

End if

It is necessary to mention here all above mechanisms developed under python language script as a development part of VXLAN based monitoring plugin for IPFIX monitoring probe YAF[90].

6.4 Flow Cache Management

Flow cache is a temporary storage place of all active network flows. Flow entries are maintained in the flow cache tables for the certain period of time. The cache size based on the expected numbers of flows to be storage, polices of flow entries expiration, flow records export and depends on the memory available in the system. The flow entries expiration and export process based on following reasons.

6.4.1 Idle flow timeout

This is timeout value of idle flow record and typically timeout value is 300 seconds. A background process continually walks through all established flow entries and compares the timestamp field with the current time and all records that have been idle for a threshold number of seconds are removed from the table to make room for new flows.

6.4.2 Active flow timeout

This is an active flow timeout value and typically active flow timeout value is 30 minutes. If the active flow record received the packet within idle time thus, this active flow will be maintained for 30 minutes in cache and after 30 minutes flow will be exported for further process and removed from the cache.

6.4.3 Natural timeout

This is based on protocol for example when a TCP flags RST and FIN observed for flow, thus the flow will be terminated.

Therefore, flow expiration polices have more impact on efficient flow cache management [149]. Figure 6.8 illustrate different expiration polices in flow cache process.

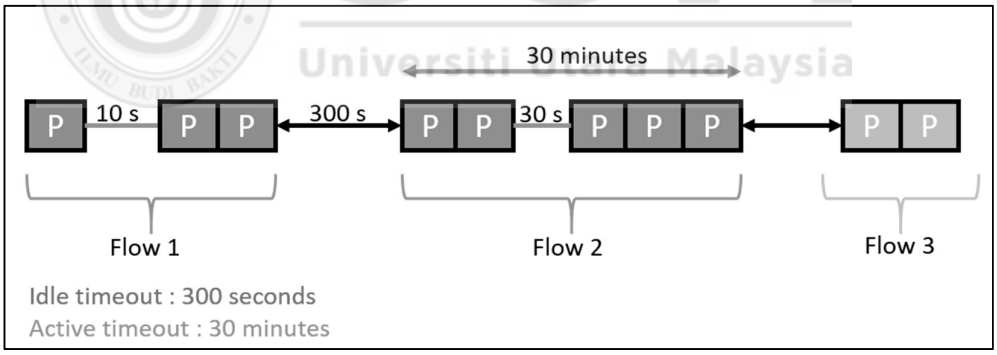

Figure 6.8. Different expiration polices in flow cache process.

The active timeout and idle timeout values have very significant impact on flow cache management and flow export process. The longer timeout values impact on number of flow entries stored in flow cache. Which is helpful to reduce the load

on flow collectors, bedside that the longer timeout values results in higher packets aggregation to be stored on flow cache, which consume higher memory. On the other hand, lower timeout values impact on storage memory of flow cache, which is helpful to reduce the memory storage, besides that the lower timeout values means higher processing load on flow export process.

6.5 IPFIX Message

A mechanism that encapsulate all flows information into records according to template set and forward these records to the collector for data analysis. This is a last step in the IPFIX processing mechanism to construct the IPFIX message in order to carry out the flow data records. However, Standard IPFIX message only support standard information elements and not able to fulfil our VXLAN based monitoring requirements. Therefore, following two new mechanisms introduced in order to enhanced current IPFIX message process for cloud overlay network monitoring. Figure 6.9 presents VXLAN based IPFIX message template record mechanisms.

A. VXLAN based Template for IPFIX Message

B. AFCM based Template for IPFIX Message

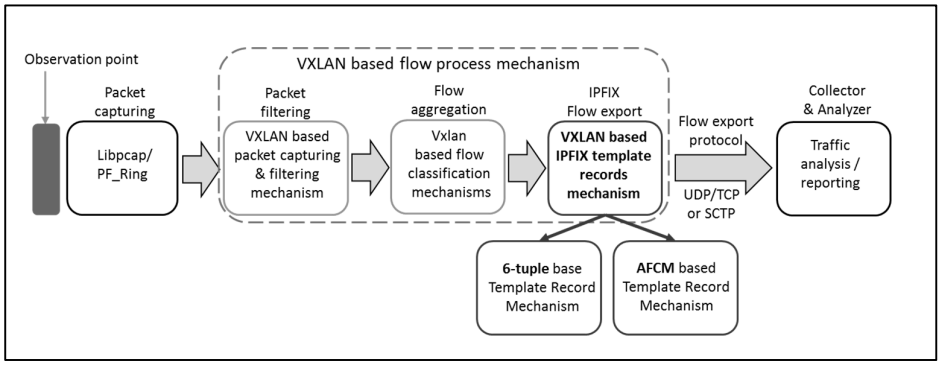

Figure 6.9. VXLAN based IPFIX Template Record Mechanisms

The simplified IPFIX message format consist of version number, message length, export time, sequence number and domain source id and different set of records. IPFIX is only open source standard which is defined in RFC7011 by IETF. The format of IPFIX message shown in Figure 6.10. The first 16 bytes of header are fixed and after that header come out with 2 bytes of template set id. Template set id used to define a layout of data records and further 2 bytes of message length define data record length. Template id value is between 0-255 and each template identify through their unique id. By default 0-1 value not used in template id, on the other side, default template value 2 used for standard IPFIX message, and 3 used for optional template. Moreover, template value id 4-255 are unassigned and may use for experimental purpose. Therefore, we selected 203 template value id for VXLAN based template and value id 204 for AFCM template for IPFIX message construction.

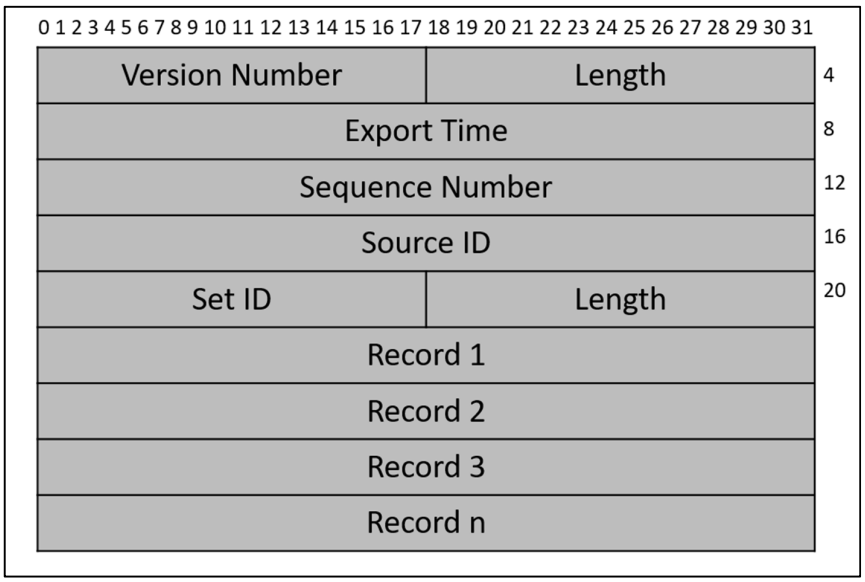

Figure 6.10. IPFIX Message Format

6.6 VXLAN based Template for IPFIX Message

The IPFIX template, based on set of fields that can be exported in flow records are named information elements [150]. The detail of IPFIX information elements explains at Internet Assigned Numbers Authority (IANA) which is responsible for maintain a standard list of IPFIX information elements [32]. The IPFIX information elements can be defined from data link layer to application layer. However common information elements are belongs to network and transport layer. On the other hand, IPFIX also support private information elements. The required VXLAN based IPFIX information elements define in Table 6.2.

Table 6.2.

VXLAN based IPFIX Information Elements

Id	Name	Description	Byte Size
152	flowStartMilliseconds	Timestamp of the flow's first packet.	8
153	flowEndMilliseconds	Timestamp of the flow's last packet.	8
8	sourceIPv4Address	IPv4 source address in the packet header.	4
12	destinationIPv4Address	IPv4 destination address in the packet header.	4
7	sourceTransportPort	Source port in the transport header	2
11	destinationTransportPort	Destination port in the transport header.	2
2	packetDeltaCount	Number of packets for the flow	8
1	octetDeltaCount	Number of bytes for the flow	8
4	protocolIdentifier	IP protocol number in the packet header	1
1001	VNILabel	VXLAN network identifier value in the packet header.	3

Except VNILabel information element, all of other information elements already define in IANA standard list of IPFIX information elements. As per our research requirement of cloud overlay network monitoring, the new 3 bytes of information element names VNILabel is added as a private information element. Moreover, the new VXLAN based IPFIX template constructed, based on Table 6.2 information elements. Hence, template ID 203 uniquely defined for VXLAN based template. Figure 6.11 presents the VXLAN based set of information elements in IPFIX message template.

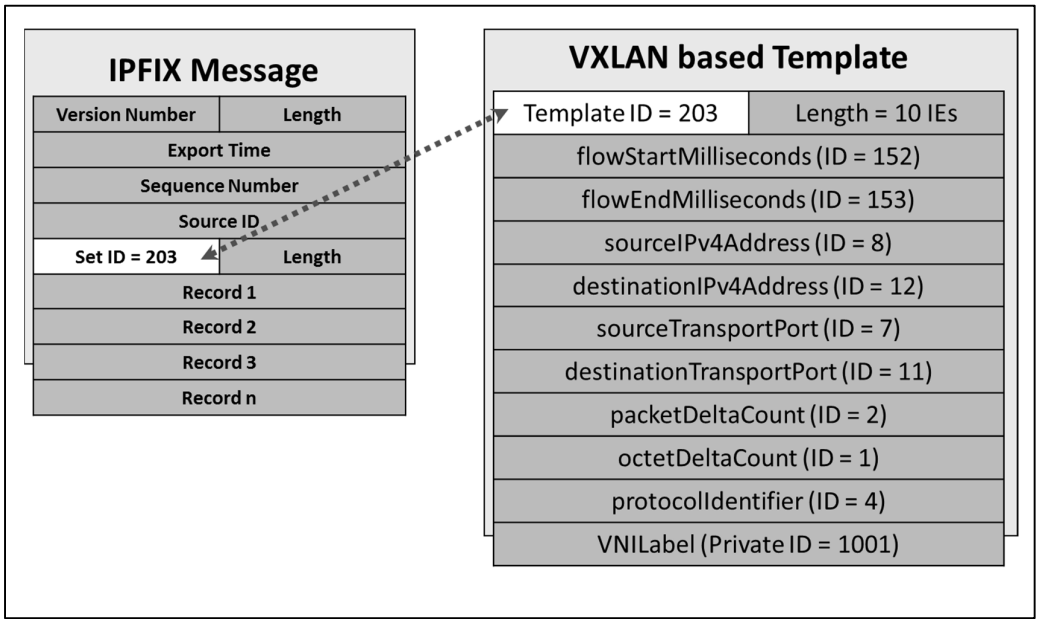

Figure 6.11. VXLAN based IPFIX Template

6.6.1 VXLAN based flow data record

Flow entries are maintained in the flow cache tables for the certain period of time. After the flow entry timeout whether it is idle or active timeout, flow data forwarded to a process for builds an IPFIX message. An IPFIX message is constructed with template ID in our case VXLAN based template and flow data are stored in IPFIX records. Moreover, data sets are used in IPFIX to carrying data records to export it to the collector. A data set is based on many different data records and each data record have flow properties based on template.

template203.py

```
import io
import operator
import functools
import struct
import ipfix.message
msg = ipfix.message.MessageBuffer()
#Vnilabel defined in IE list as pen (private element number)
import ipfix.ie
import ipfix.template
class Template:
    def __init__(self, tid = 203):
        self.tid = tid
        self.ies = [152,153,8,12,7,11,2,1,4,1001]
        if iterable:
            if not isinstance(iterable, ie.InformationElementList):
                iterable = ie.InformationElementList(iterable)
            for elem in iterable:
                self.append(elem)
    def __repr__(self):
        return "<Template ID "+str(self.tid)+" count "+ \
               str(len(self.ies))">"
    def count(self):
               return len(self.ies)
    def fixlen_count(self):
               return self.count()
#Message record for template
class Message:
tmpl = ipfix.template.from_ielist(203,
...         ipfix.ie.spec_list(flowStartMilliseconds,
...                            flowEndMilliseconds,
...                            sourceIPv4Address,
...                            destinationIPv4Address,
...                            sourceTransportPort,
...                            destinationTransportPort,
...                            packetDeltaCount,
...                            octetDeltaCount,
...                            protocolIdentifier,
...                            vnilabel))
msg.add_template(tmpl)
    def finalize(self):
        self.packplan = TemplatePackingPlan(self, range(self.count()))
    def packplan_for_ielist(self, ielist):
        return TemplatePackPlan(self, [self.ies.index(ie) for ie in ielist])
msg.export
```

Figure 6.12. VXLAN based Flow Record Mechanism

Codes written in python depicted in Figure 6.12 for template id 203 and set of information elements. A 48-bytes required for construction of VXLAN based flow record in IPFIX message. Moreover, VXLAN based template set consist of 10 information elements in length as compare to standard IPFIX template set consist of 12 information elements and carry less traffic as compare to standard IPFIX

template data records. Figure 6.13 presents multiple flow records in the IPFIX message and also presents the VXLAN based flow data record.

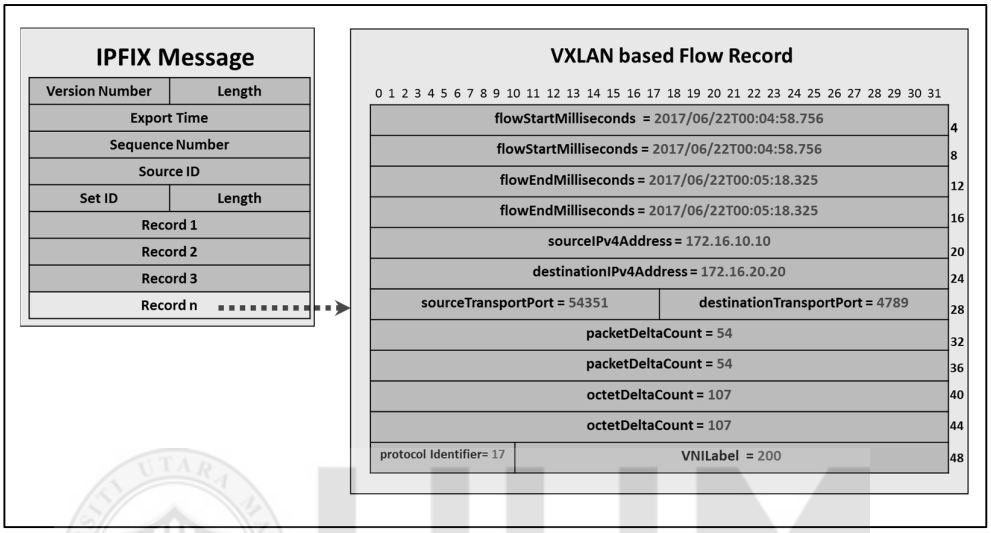

Figure 6.13. VXLAN based Flow Record in IPFIX message

6.7 AFCM based Template for IPFIX Message

A template defined complete structure and set of data in order to carry data with IPFIX message to communicate with collector. After the expiration of either idle or active or natural timeout in flow cache, AFCM based summarized information flows forwarded to next stage to construction of IPFIX message or use standard IPFIX message template. As per our research requirement standard IPFIX message not fulfil our requirements. Therefore, we designed new template with unique template ID 204, using python script in python language a newly AFCM template with set of exclusive seven information elements IPFIX message constructed, based on adaptable flow classification mechanism key and non-key fields. Furthermore, data type, length and id specified each of element which is part of

AFCM based template including private information element VNILable, id and length. The newly constructed AFCM based IPFIX information elements define in Table 6.3.

Table 6.3.

AFCM based IPFIX Information Elements

Id	Name	Description	Byte Size
152	flowStartMilliseconds	Timestamp of the flow's first packet.	8
153	flowEndMilliseconds	Timestamp of the flow's last packet.	8
8	sourceIPv4Address	IPv4 source address in the packet header.	4
12	destinationIPv4Address	IPv4 destination address in the packet header.	4
2	packetDeltaCount	Number of packets for the flow	8
1	octetDeltaCount	Number of bytes for the flow	8
1001	VNILabel	VXLAN network identifier value in the packet header.	3

After summing-up aforementioned information 6 Out of 7 information elements defined in Internet Assigned Numbers Authority (IANA) and value id from 1-482 already pre-assigned to different information elements. However, value id 483-32767 information elements are unassigned and may use for experimental purpose or defined as private information element. Hence, we used value id 1001 as private information element and defined data type VNI Lable information as per our research requirement and assigned integer value unsigned int for 4 bytes as VNI

value range between 1 to 16 million. Figure 6.14 presents the AFCM based set of information elements in IPFIX message template.

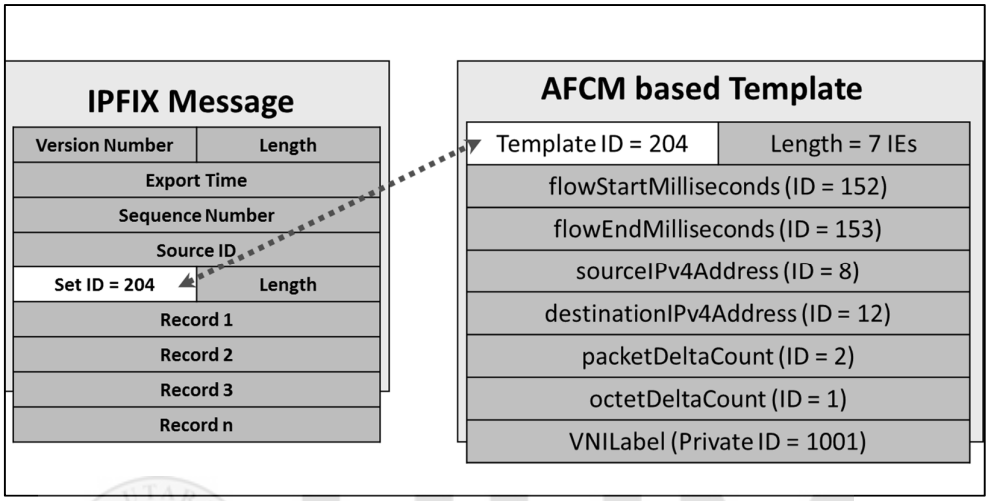

Figure 6.14. AFCM based IPFIX Template

6.7.1 AFCM based flow data record

Once AFCM based flows expired in the cache flow then AFCM summarized flow information forward to builds an IPFIX message. An IPFIX message is constructed with template ID 204 and flow data is enclosed in data records based on AFCM template set information elements. A 43-bytes required for construction of AFCM based flow record in IPFIX message. AFCM based template set consist of 7 information elements in length as compare to VXLAN based template set consist of 10 information elements and standard IPFIX template set consist of 12 information elements. Hence, AFCM based IPFIX

message flow records carry less traffic as compare with standard IPFIX message and VXLAN based IPFIX message flow records.

AFCM-template.py

```
import io
import operator
import functools
import struct
import ipfix.message
msg = ipfix.message.MessageBuffer()
#Vnilabel defined in IE list as pen (private element number)
import ipfix.ie
import ipfix.template
class Template:
    def __init__(self, tid = 204):
        self.tid = tid
        self.ies = [152,153,8,12,2,1,1001]
        if iterable:
            if not isinstance(iterable, ie.InformationElementList):
                iterable = ie.InformationElementList(iterable)
            for elem in iterable:
                self.append(elem)
    def __repr__(self):
        return "<Template ID "+str(self.tid)+" count "+ \
               str(len(self.ies))">"
    def count(self):
                return len(self.ies)
    def fixlen_count(self):
                return self.count()
#Message record for template
class Message:
tmpl = ipfix.template.from_ielist(204,
...         ipfix.ie.spec_list(flowStartMilliseconds,
...                            flowEndMilliseconds,
...                            sourceIPv4Address,
...                            destinationIPv4Address,
...                            packetDeltaCount,
...                            octetDeltaCount,
...                            vnilabel))
 msg.add_template(tmpl)
    def finalize(self):
        self.packplan = TemplatePackingPlan(self, range(self.count()))
    def packplan_for_ielist(self, ielist):
        return TemplatePackPlan(self, [self.ies.index(ie) for ie in ielist])
msg.export
```

Figure 6.15. AFCM based Flow Record Mechanism

A python script developed in python language depicted in Figure 6.15 for construction of AFCM based template for IPFIX message with set of exclusive seven information elements including key and non-key fields and embedded with

plugin. Figure 6.16 presents multiple flow records in the IPFIX message and also presents the AFCM based flow data record.

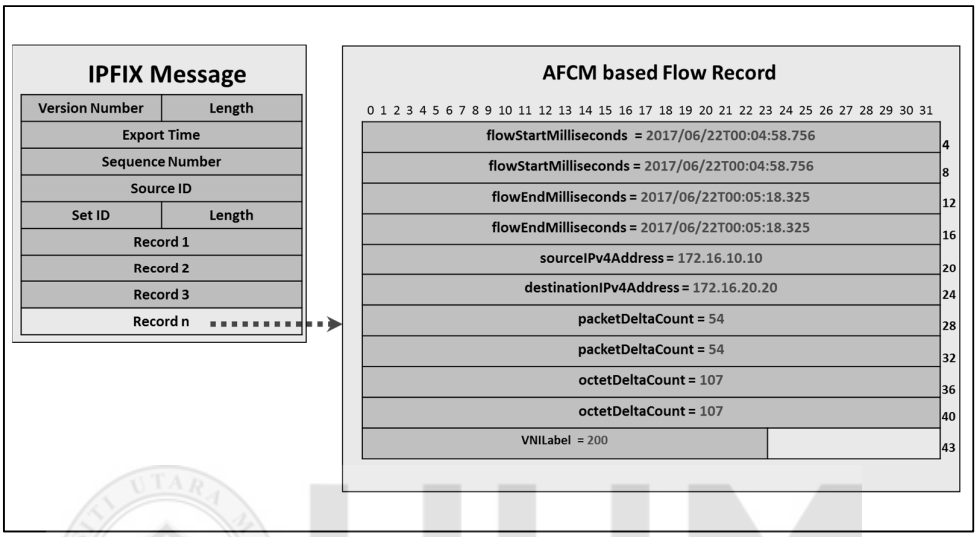

Figure 6.16. AFCM based Flow Record in IPFIX message

6.8 Flow Export Process

The flow export process defines how to carry VXLAN based IPFIX messages via multiple transport protocols from flow export process to Flow collector for further data analysis. IPFIX has ability to support multiple transport protocol for flow export [136]. IPFIX messages can be transferred using UDP, TCP, SCTP or IPFIX file format. After the construction of VXLAN based IPFIX message and AFCM based IPFIX message, UDP has been selected as a transport protocol for export the flow records to flow collector. As UDP carried no overhead and it is widely deployed transport protocol for flow export process.

6.9 Flow Collection and Traffic Analysis

The flow collector responsible for receiving flow data which is created and exported by flow exporter for storage, archive and organized them into uniform format for further analysis. It is working like a reception and received data from multiple flow exporters and store according to the requirement for further network traffic performance analysis. Analysis of network traffic in a high speed networks are one of the challenging task. In large and virtualized cloud networks where every packet has to capture and decode, and the volume of the data is huge for analysis that will have an impact on performance of analysis. Analyzer allows deep into the network to see where threats are originating and how the network is being used. The selection for the analyzing tool, it should be compatible to cooperate with the different flow exporter and process data from several distributed flow exporters. Furthermore, it should be presented and visualize the required data in well-organized form so that every user should be able to get the wanted information for further decision. There are many commercial and open source flow collector and analyzer available in the market. To the selection of flow collector and analyzer, it must be able to understand the IPFIX format sampled data. Similarly, it should be supported and run on wide-spread operating system. For this purpose we selected SiLk [105] open source flow collector and network traffic analyzer since it has compatibility with YAF exporter. It has ability to near real time traffic analysis with high performance data logging and graphing for long run traffic analysis. Since it is an open source tool and understand IPFIX message sampled data and supported all transport protocols.

6.10 Simulation and Experiment Results with Performance Analysis

Simulation performed on Linux based virtual environment, using Mininet [127] simulation tool, virtual machines, OpenVswitches [107] and different network segments created for cloud overlay networks environment. Traffic generated through well-known network tool iperf [112] between different virtual machines for monitoring performance measurement in different conditions. A plugin is developed and compiled with an open source tool YAF [103] based on proposed algorithms to enhanced existing IPFIX flow monitoring mechanisms for VXLAN based cloud overlay networks. Refer to the Figure 4.1, traffic generated between different network segments using benchmarking methodology with transmission duration 60 minutes using the dataset presented in Table 4.1.

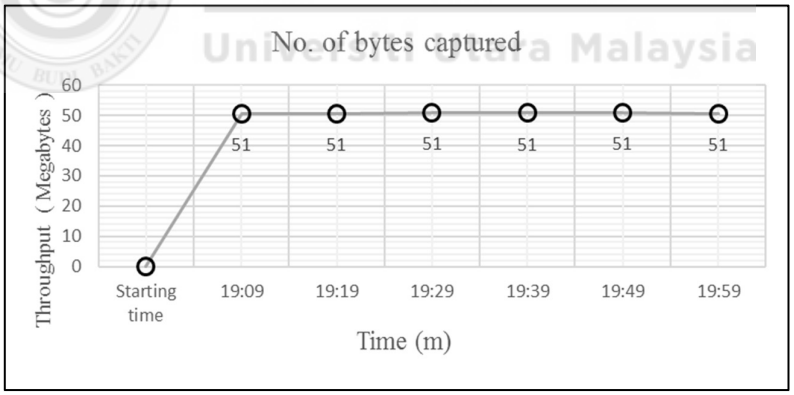

Figure 6.17. Standard Flow Monitoring based on 5-tuple (64 Kbps)

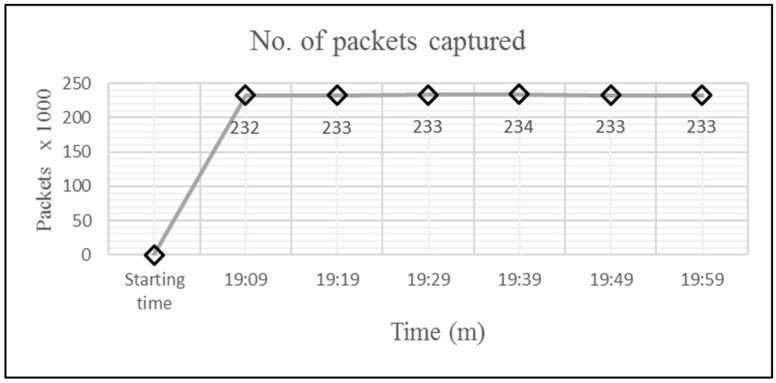

Figure 6.18. Standard Flow Monitoring based on 5-tuple (64 Kbps)

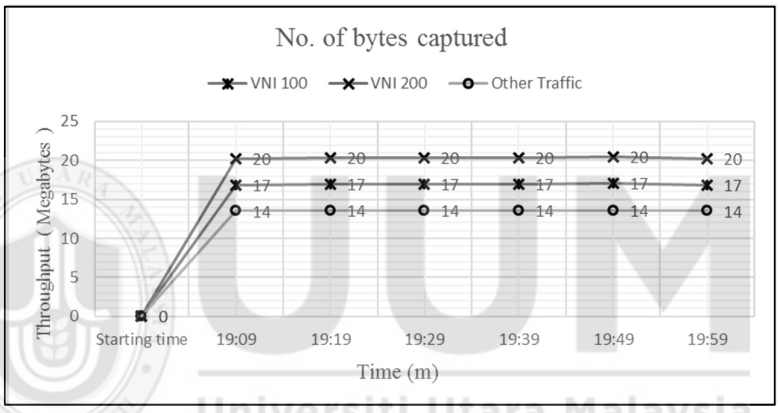

Figure 6.19. Enhanced IPFIX Flow Monitoring detail (64 Kbps)

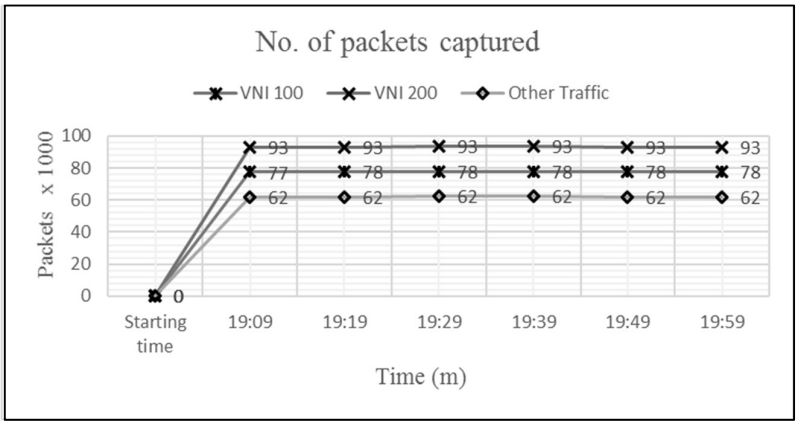

Figure 6.20. Enhanced IPFIX Flow Monitoring detail (64 Kbps)

The experiment results shown in Figure 6.17 and 6.18 with standard monitoring tool which is only capture the total number of packets and bandwidth but could not identify the VXLAN based tunnel traffic in virtual cloud network environment. On the other hand, Figures 6.19 represents the results of traffic captured in packets between different virtual machines separately, while Figure 6.20 illustrates the bandwidth captured in bytes between different virtual machines separately with total of 64 Kbps traffic load. The results presented evidence of that our proposed enhanced IPFIX monitoring system, which is able to captured VXLAN packets and differentiate the traffic based on Virtual Network Identifier (VNI) and other traffic. In addition to this traffic, packet delay calculated as follows:

$$\text{Packet Delay} = \text{Propagation Delay} + \text{Serialization Dealy} + \text{Queuing Delay} + \text{Processing Delay}$$

The propagation delay given by

$$\text{Propagation Delay} = \frac{\text{distance (meter)}}{time\ (seconds)} = \frac{d}{s} * \frac{1}{n} \tag{6.1}$$

where 1/n is velocity factor of medium, *d* is distance in meters and *s* presents time in seconds.

The Serialization delay derived by

$$\text{Serialization Delay} = \frac{\text{Packet\_Size (Number of bytes)}}{\text{bandwidth(seconds)}} = \frac{S}{B} \tag{6.2}$$

where S is packet size in bytes and B is available bandwidth.

The queue delay is given by

$$\text{Queuing Delay} = \frac{\text{No of packets} * \text{Size of packets}}{2 * \text{bandwidth}(seconds)} = \frac{N * S}{2 * B} \tag{6.3}$$

where N is number of packets.

Processing Delay = usually in milliseconds therefore it is negligible

The approximate estimates of the total delay

$$\text{Total Packet Delay} = \frac{d}{s} * \frac{1}{n} + \frac{S}{B} + \frac{N * S}{2 * B} + 1\text{milisecond} \tag{6.4}$$

Therefore

$$\text{Total Packet Delay} = \frac{d}{s} * \frac{1}{n} + \frac{S}{B} + \frac{N * S}{2 * B} \tag{6.5}$$

Equation 6.5 is derived total packet delay on the network path.

Furthermore, to measure the packet loss network performance metric. All Packets were captured during simulation and stored into two variables TxPackets and RxPackets between two nodes then calculate the packet loss and packet loss ratio.

$$\text{Loss Packets} = \text{TxPackets} - \text{RxPackets} \tag{6.6}$$

The Packet loss ratio calculated as follows:

$$\text{Packet loss ratio} = \frac{\text{Loss Packets}}{\text{TxPackets}} \tag{6.7}$$

where TxPackets are total number of packets transmit over the link and RxPackets are total number of packets received.

Since the simulation performed in lab environment hence, there is no packet delay and packet loss over the network link.

The throughput calculated as follows on network link:

$$\text{Throughput} = \frac{\text{RW}}{\text{RTT}} \tag{6.8}$$

where RW is the TCP Receive Window and RTT is the round-trip time for the network path.

Moreover, the results given in Figure 6.19 and 6.20, VNI 100 represent the captured tunnel traffic between virtual machines A1 and A2 refer to Figure 4.1, similarly VNI 200 represent the captured tunnel traffic between virtual machines B1 and B2 with time and date stamp duration, and remaining traffic shown as other traffic. Hence, proposed enhanced IPFIX mechanism seems to be improved monitoring mechanism in performance as compared to standard IPFIX monitoring systems. The above results are based on following the benchmark methodology with 64 Kbps traffic of dataset. Likewise, another simulation performed with the same environment to evaluate the performance with 1 Mbps dataset as given in Table 4.1.

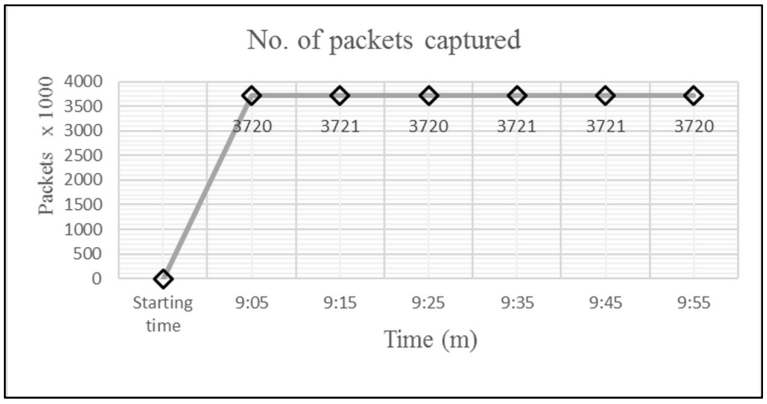

Figure 6.21. Standard IPFIX Monitoring based on 5-tuple (1 Mbps traffic)

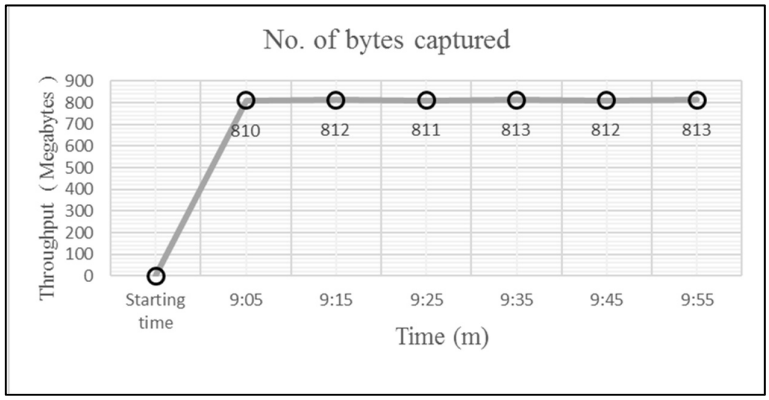

Figure 6.22. Standard IPFIX Monitoring based on 5-tuple (1 Mbps traffic)

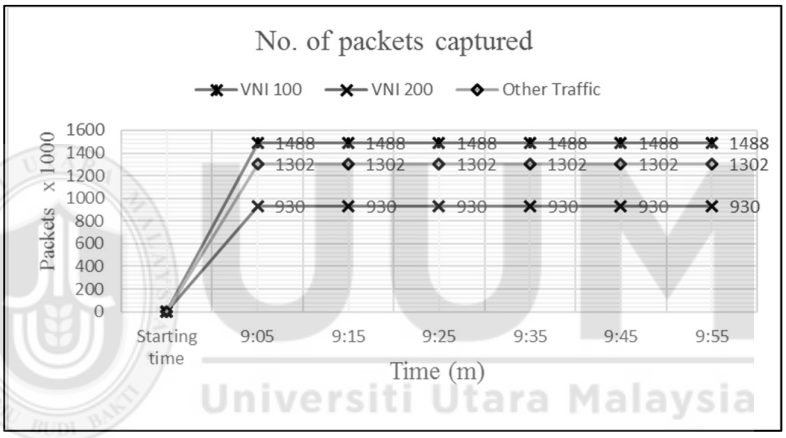

Figure 6.23. Enhanced IPFIX Flow Monitoring results (1 Mbps traffic)

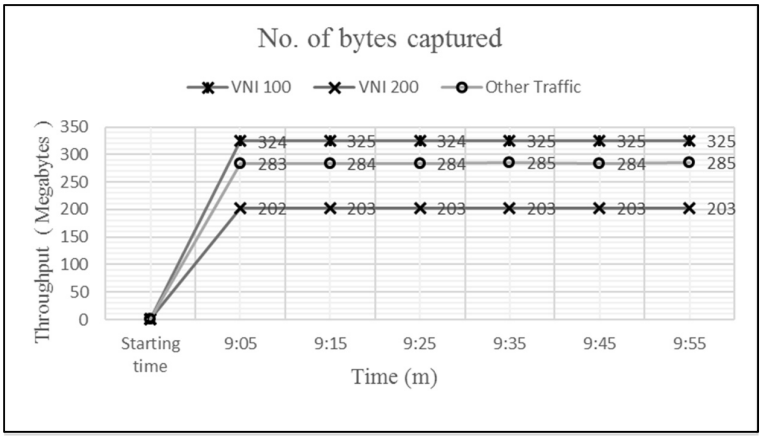

Figure 6.24. Enhanced IPFIX Flow Monitoring results (1 Mbps traffic)

The experiment results shown in Figure 6.21 and 6.22 with standard IPFIX monitoring tool which is only able to capture the total number of packets and bandwidth but could not identify the VXLAN based tunnel traffic in virtual cloud network environment. Whereas, Figure 6.23 and 6.24 represents the VNI 100 tunnel traffic captured between virtual machines A1 and A2 in the shape of bytes and packets separately, similarly VNI 200 tunnel traffic captured between virtual machines B1 and B2, and other traffic captured between virtual machines C1 and C2, refer to cloud overlay network environment Figure 4.1. It can be seen that results shows as per expected of proposed enhanced IPFIX monitoring system, which is able to captured VXLAN packets and differentiate the traffic based on Virtual Network Identifier (VNI) and other traffic. However, standard IPFIX monitoring tool not able to captured the virtual tunnel traffic and only shows as total traffic.

Furthermore, it is imperative to evaluate the CPU processing load for our proposed enhanced IPFIX processing monitoring mechanism. The CPU processing load data were collected with same simulation environment and presented in Table 6.4 for 64 Kbps traffic load and in the Table 6.5 for 1 Mbps traffic load, in order to compare the results with standard IPFIX processing load as mention in Section 4.6 in Chapter 4.

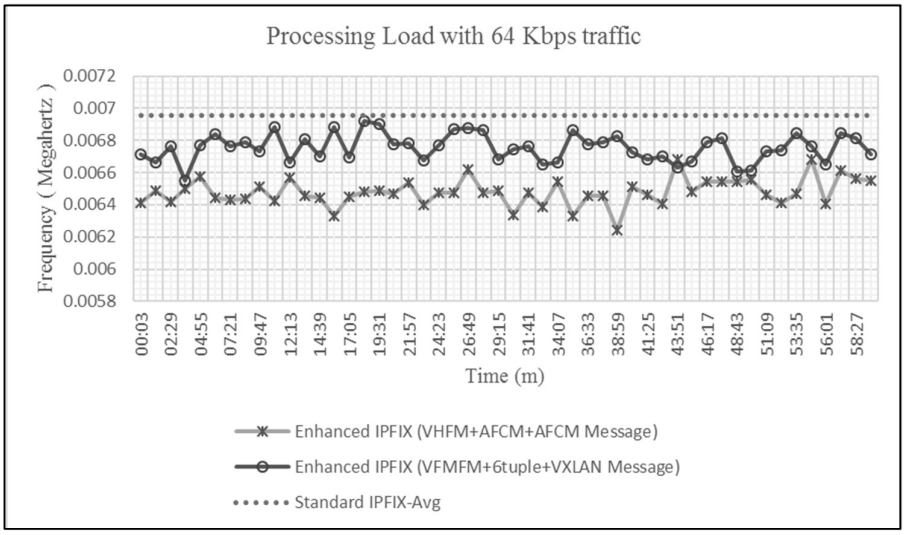

Figure 6.25. Processing load of Enhanced IPFIX Mechanisms (64 Kbps)

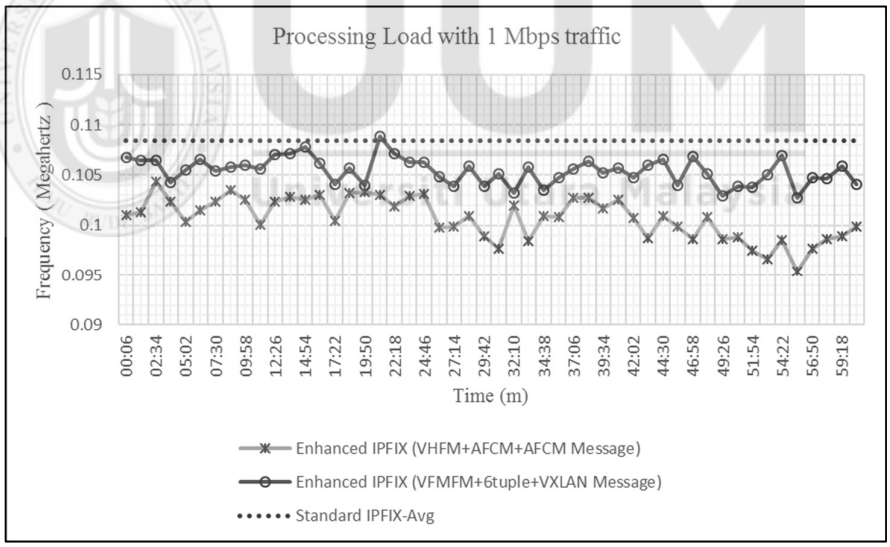

Figure 6.26. Processing load of Enhanced IPFIX Mechanisms (1 Mbps)

Figure 6.25 and 6.26 present proposed enhanced IPFIX mechanisms processing load with 60 minutes simulation on 64 Kbps traffic and respectively 1 Mbps traffic

load along with the combination of proposed mechanisms for all three processing stages on enhanced IPFIX. It can be seen that monitoring mechanisms with the combination of enhanced IPFIX (VHFM+AFCM+AFCM Message) and (VFMFM+6tuple+VXLAN Message) took less processing load as compare to average IPFIX processing load. However, to ensure the results are accurate, same simulation with same environment repeated 10 times to calculate the average processing load of both enhanced IPFIX mechanisms. CPU Processing data were collected and presented in Table 6.4 with 64 Kbps traffic.

Table 6.4.

Enhanced IPFIX processing load collected in MHz with 64 Kbps traffic.

No of test	Enhanced IPFIX (VFMFM+6tuple+VXLAN Message)	Enhanced IPFIX (VHFM+AFCM+AFCM Message)
1	0.00675481	0.006478068
2	0.00651164	0.006244857
3	0.00674130	0.006044037
4	0.00668727	0.006238379
5	0.00666700	0.006063471
6	0.00661972	0.006160642
7	0.00666025	0.006031081
8	0.00674806	0.006231901
9	0.00655217	0.006056993
10	0.00655217	0.006270769
Avg. Load	**0.00664944**	**0.006182027**

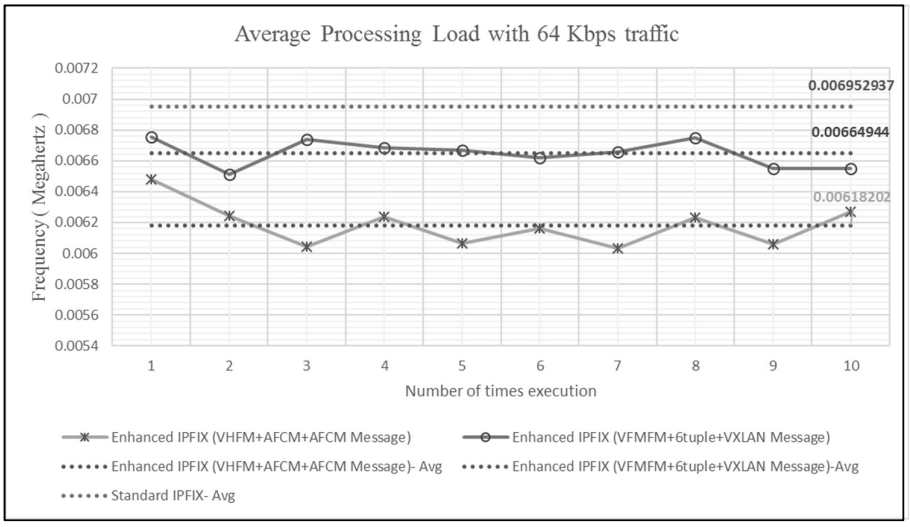

Figure 6.27. Average Processing load of Enhanced IPFIX Mechanisms (64 Kbps)

Figures 6.27 and 6.28 shows average processing load of both enhanced IPFIX (VHFM+AFCM+AFCM Message) and (VFMFM+6tuple+VXLAN Message) monitoring mechanisms with 64 Kbps traffic.

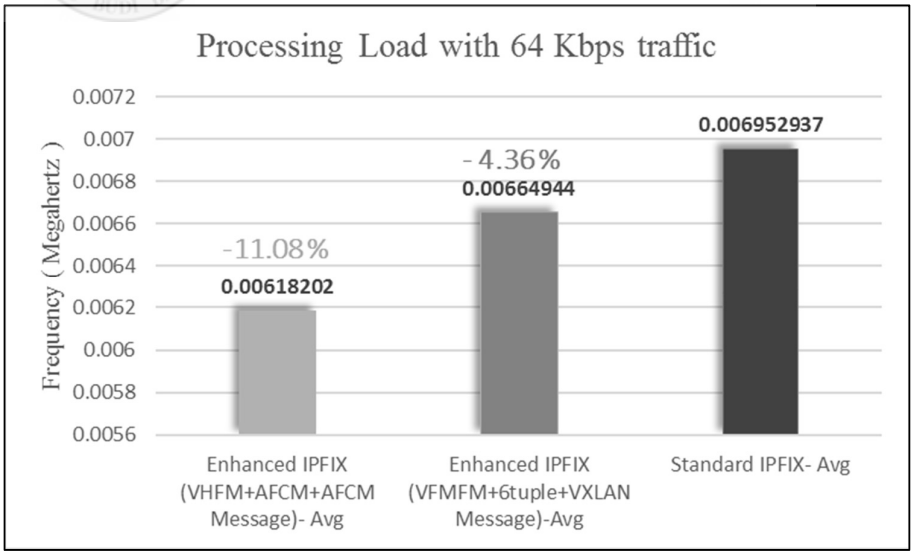

Figure 6.28. Enhanced IPFIX processing load with 64 Kbps traffic

It can be seen that combination of (VHFM+AFCM+AFCM Message) mechanisms of enhanced IPFIX took on average 11.08% percent less processing load as compare to average IPFIX processing load, while (VFMFM+6tuple+VXLAN Message) mechanisms of enhanced IPFIX took 4.36% percent less load as compare to IPFIX. Similarly, CPU Processing data with 1 Mbps traffic were collected and presented in Table 6.5.

Table 6.5.

Enhanced IPFIX processing load collected in MHz with 1 Mbps traffic.

No of test	Enhanced IPFIX (VFMFM+6tuple+VXLAN Message)	Enhanced IPFIX (VHFM+AFCM+AFCM Message)
1	0.10543154	0.100755371
2	0.10437722	0.099747817
3	0.10226859	0.092694941
4	0.10258488	0.092392675
5	0.10543154	0.097732717
6	0.10290118	0.097732713
7	0.10226859	0.095012315
8	0.10437722	0.093702495
9	0.10216316	0.096725156
10	0.10237402	0.093702495
Avg. Load	**0.103417793**	**0.096019868**

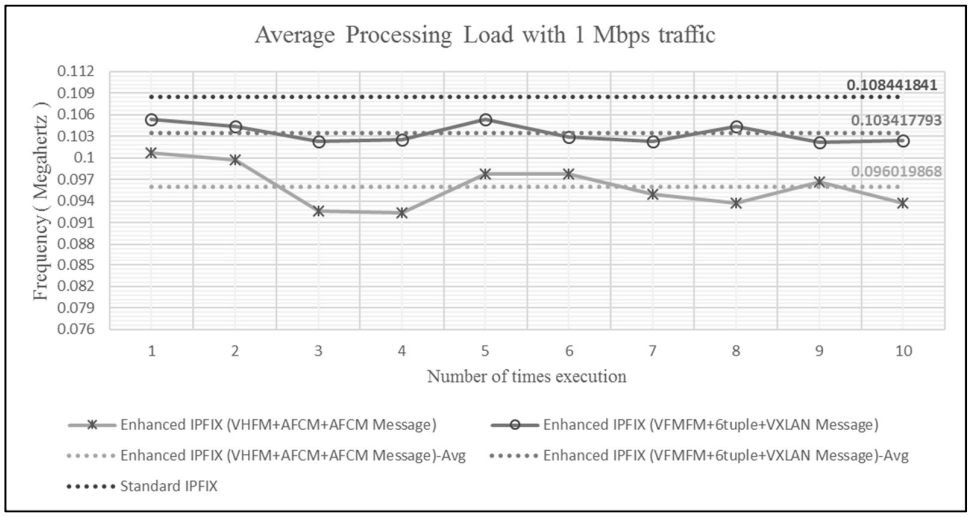

Figure 6.29. Average Processing load of Enhanced IPFIX Mechanisms (1 Mbps)

Similarly Figures 6.29 and 6.30 presents average processing load of proposed mechanisms with 1 Mbps traffic. It can also be seen that combination of (VHFM+AFCM+AFCM Message) mechanisms of IPFIX took overall 11.45% percent less CPU utilization as compare to average standard IPFIX CPU utilization. Whereas (VFMFM+6tuple+VXLAN Message) mechanisms of

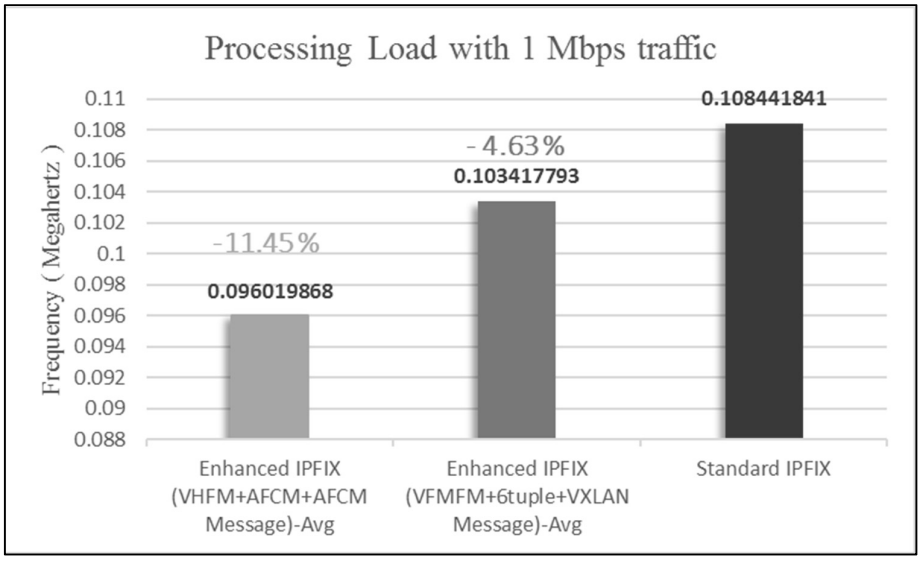

Figure 6.30. Enhanced IPFIX processing load with 1 Mbps traffic

enhanced IPFIX took 4.63% percent less as compare to standard IPFIX but slightly high CPU utilization as compare to (VHFM+AFCM+AFCM Message) mechanisms.

Following the benchmarking methodology results were obtained from number of tests performed. The simulation results were obtained on Intel 1.8GHz Quad core CPU with 2 GB RAM based Linux machine running with network probe. In other words, average processing load is benchmark of our proposed enhanced IPFIX mechanisms.

The presented results and studies have been evaluated and it can be seen that both combination of enhanced IPFIX mechanisms have faster performance than standard IPFIX and consume less processing load as compare to standard IPFIX. Furthermore, experiment results have been confirmed that proposed mechanisms able to captured the live tunnel traffic and also clearly identify VXLAN packets and distinguish the traffic between virtual machines and other traffic in high speed cloud virtual network environment. Hence, proposed enhanced IPFIX mechanisms seems to be improved monitoring mechanism in performance as compared to standard IPFIX monitoring systems.

Proposed mechanisms will help in depth performance analysis and troubleshooting of cloud overlay multi-tenant networks issues. It would help multiple stakeholders related to the cloud computing including consumers, enterprises and cloud service providers.

6.11 Summary

This chapter introduced flow classification and IPFIX template record mechanisms to process and identify the VXLAN based overlay network traffic. The proposed mechanisms can capture the invisible cloud overlay network traffic to identify, track, analyze and monitor the performance of cloud overlay network services. A plugin developed and compile with open source monitoring software for cloud overlay network monitoring system, as the performance of a system is dynamic and depends on multiple parameters, the proposed system is capable of continuously tracking, quantifying and updating the results. The experiment performed on Linux based virtual cloud environment and compare the results with standard IPFIX monitoring system and proposed enhanced IPFIX monitoring system. The proposed monitoring system clearly identify the invisible VXLAN based overlay network traffic and also differentiate the VXLAN based LAN segments traffic, while proposed monitoring system faster than standard IPFIX system. The proposed monitoring system can real time track and monitor the cloud overlay network traffic and also can be used in depth performance analysis and troubleshooting of cloud overlay networks issues in large cloud network infrastructure. Moreover, proposed mechanisms act faster and consume less processing load as compare to standard IPFIX, combination of (VHFM+AFCM+AFCM Message) mechanisms of enhanced IPFIX took on average 11.08% percent less processing load, while (VFMFM+6tuple+VXLAN Message) mechanisms of enhanced IPFIX took 4.36% percent less load as compare to standard IPFIX with 64 Kbps traffic load. Similarly, combination of

(VHFM+AFCM+AFCM Message) mechanisms of enhanced IPFIX took overall 11.45% percent less processing load, whereas combination of (VFMFM+6tuple+VXLAN Message) mechanisms of enhanced IPFIX took 4.63% percent less as compare to standard IPFIX. Hence, proposed system seems to be improved monitoring mechanism in performance as compared to standard IPFIX monitoring systems.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

The main goal of this research project is to design a real time cloud overlay network monitoring and performance analysis mechanism for large cloud infrastructure. The previous chapters elaborated the research conducted and the mechanisms developed in detail along with the detailed description on the literature review carried out and the methodology adopted. This chapter presents the conclusion of the research in terms of highlighting the important point, contributions, limitations of the work and recommendations for future work.

The organization of this chapter is as follows. Section 7.1 provides the brief introduction to the chapter along with its organization. Section 7.2 summarizes the work carried out highlighting the important points. Section 7.3 presents the contributions of this research, while section 7.4 explains the limitations of the work. Finally, section 7.5 offers same suggestion for future work that provides an indication to the directions in which this work can be further developed.

7.1 Summary of Research

The cloud services are rapidly growing day by day due to fast adoption and migration of workload from private data centers to cloud data centers. In addition, cloud data centers should handle higher traffic load due to this workload migration [7]. In cloud environment service providers use virtualization and automation for

cloud services. Therefore, cloud data centers required efficient cloud network traffic handling for higher capacity, increased performance and great throughput.

Cloud network plays important role to migrating and storing the workload from private data center to cloud data centers. Cloud traffic is expected 33% grow by annual growth rate (Compound Annual Growth Rate, CAGR) and by 2019 86% of workloads will be processed by cloud data center while only 14% will be processed by traditional data centers with 8% CAGR [16]. Due to dynamic resource provisioning and high speed virtualized cloud networks, growing of cloud network traffic also creating problem to manage it efficiently by service provider and end user prospectus. Therefore, cloud network performance monitoring system is required to monitor, troubleshoot, and analyze what is happening across your enterprise network environment. Therefore, it is highly recommended to quickly and proactively resolve any network-based performance issues with end-to-end visibility and actionable insights.

Network virtualization refers to the combination of available physical network resources using hypervisor software into a single virtual network [37]. Available bandwidth divided into different virtual secure channels and that may assigned to any virtual machine in a cloud computing environment. Network virtualization allows running of isolated logical networks on a shared physical network. It consists of a combination of multiple network resources, capabilities and functionalities into a single unit known as a virtual network. It is the solution for expanding data center devices that connect each other within virtualized environment.

Overlay network allow cloud providers and end users to orchestrate networks along with other virtual resources in cloud environment. It provides new path to converged network and run as independently virtual network on top of physical network. The cloud overlay network allows network resources to be dynamically provisioned similarly to virtual storage and virtual compute. Cloud overlay network technology is used in cloud data centers, to effectively isolate multiple tenants and automate network-wide virtual machine migration that fully satisfy the requirements of large cloud service providers and enterprises.

Cloud overlay technology introduces the same visibility challenges as most exist for encapsulation methods. Essentially, end-to-end traffic is hidden inside the tunnel, so it must be able to strip away the encapsulation for sustained monitoring and troubleshooting. Currently overlay network technology in cloud infrastructure have visibility gaps, which mean cloud provider and consumers can miss out the major performance issues for troubleshooting of overlay network traffic. Thus, to keep a close watch on network and catch potential problems, an urgent need of network monitoring tool to dynamically track and report more in-depth for not only see the invisible traffic but also presents the related information of cloud overlay network technologies.

Cloud monitoring is an undertaking grade solution that helps to maintain applications active and performing well constantly. A multi-tenant nature of cloud can be challenging for smooth management in terms of performance constraints and quality of service because the services of cloud are scalable, flexible and on demand. Monitoring can play important role in utilization of cloud resources of all

layers. Cloud monitoring is essential for both cloud users and provider. Along with, it is a central tool for managing software and hardware infrastructure. Moreover, it furnishes information and key performance indicators for cloud layer services.

This research concentrates on filling this gap of monitoring in the cloud overlay network environment. In this regard, this research developed mechanisms that could quantify and differentiate of cloud overlay network traffic between different virtual machines to single cloud to multiple cloud network environment. The proposed enhanced monitoring architecture of the system consists of several stages, that includes overlay packet observation and selection mechanism, enhanced flow classification mechanism and enhanced IPFIX messages mechanism for cloud overlay network monitoring and traffic analysis.

The overlay packet observation and selection mechanism developed in this research is capable to enhance the existing packet capturing system and add new technique to inspect every captured packet and select only VXLAN packets. Since the cloud overlay is a new technology and packets are encapsulated thus the most critical step is retrieving and sampling the VXLAN packets. Packets are inspected based on header instead of whole payload inspection to reduce the overhead and minimize the load of packet selection stage. Thus, selected packet becomes an element of the output packet stream. The mechanisms developed have been tested on testbed and the simulation results showed that proposed mechanism functionality working as expected.

The enhanced flow classification mechanism introduces the new flow pattern called VXLAN based 6-tuple flow, which represents set of six field values instead of

typical 5- tuple field values. The newly added VNI key field on traditional flow key pattern which makes its unique 6-tuple VXLAN based flow pattern. After the filtering each packet that arrives in the flow classifier the relevant 6-tuples header fields are extracted, and mapped the packet into the existing flow or new flow based on cache flow entries. Flow classification is used to map each input packet to the flow it belongs to. Moreover, to further reduce the data generated by network probes, An Adoptable Flow classification mechanism has been introduced, the proposed mechanism based on defined rules, and only aggregate the flows based on given and required flow key fields. The proposed mechanism has ability to reduce the amount of monitoring data before exporting to collector for data analysis, and also help to reduce the CPU utilization as compare to standard IPFIX process.

The last part of mechanism developed in this research is to introduce VXLAN based and AFCM based message templates for data records used in IPFIX monitoring architecture. Flow entries are maintained in the flow cache tables for the certain period of time. After the flow entry timeout whether it is idle or active timeout, flow data forwarded to a process for builds an IPFIX message. A 48-bytes VXLAN based IPFIX message template constructed with template ID 203 to store flow data in the records. Similarly, AFCM based 43-bytes IPFIX message constructed with template ID 204 to store the flow records. Finally, these records forwarded via multiple transport protocols from flow export process to Flow collector for further data analysis. The experiment performed and results show clearly identify the invisible VXLAN based overlay network traffic and also differentiate the VXLAN

based Lan segments traffic as compare with standard monitoring systems. Furthermore, proposed mechanism consume less processing power of CPU and the performance is faster than standard IPFIX system.

All the proposed mechanisms were tested for their functionality and validated by comparing with other standard monitoring systems using simulations. Due to lack of resources, a hybrid technique has been used in this research project with combination of testbed and network simulation tools. The simulation environment consists of cloud controller, network controller, mininet emulator used on top of Open vSwitches on the hypervisors using OpenFlow to establish overlay networks. The experiments carried out show that the proposed monitoring system outperform all the other monitoring system proposed in the literature.

Overall the proposed monitoring system can real time track and monitor the cloud overlay network traffic and also can be used in depth performance analysis and troubleshooting of cloud overlay networks issues in large cloud network infrastructure. Moreover, proposed system consume less power processing of CPU. Finally, a plugin has been developed which can be compile with open source monitoring software for cloud overlay network monitoring system.

7.2 Research Contribution

The overall contribution of this research is development of cloud overlay network monitoring system and environment with enhanced IPFIX flow monitoring architecture. The proposed system can capture the invisible cloud overlay network traffic to identify, track, analyze and monitor the performance of cloud overlay

network services. Proposed monitoring system can provide network operators with detailed information about the traffic traversing a linked and related information specifically suited to the modern cloud-scale data center. It would help cloud network operators and users to quickly and proactively resolve any overlay network based performance issues with end-to-end visibility and actionable insights. The specific contributions of this research are listed below:

1. Enhanced packet observation and selection mechanism for cloud overlay network monitoring.
    a) The development of VXLAN Field Match Filtering Mechanism (VFMFM)
    b) The development of VXLAN based Hash Filtering Mechanism (VHFM)
2. Enhanced flow classification mechanism for cloud overlay network monitoring.
    a) The development of VXLAN based 6-tuple Flow Classification Mechanism.
    b) The development of VXLAN based Adoptable Flow Classification Mechanism (AFCM).
3. Enhanced IPFIX messages mechanism for cloud overlay network monitoring.
    a) The development of VXLAN based flow template record with in IPFIX message mechanism
    b) The development of VXLAN based AFCM flow template record with in IPFIX message mechanism

4. Design and development of VXLAN based cloud overlay network environment for IPFIX performance benchmarking.
5. The development of VXLAN based plugin to real time monitoring of cloud overlay networks for open source monitoring software.

7.3 Research Limitations

Although this research has successfully developed, tested and evaluated under emulated condition. Despite the emulation environment been setup closely resemble with real world conditions, all the uncertainties that can be experienced in the real world may not be included into the emulation environment. Though it has theoretically and as well as practically been shown that proposed mechanisms can incorporate any standard monitoring system. The limited resources such as bandwidth of links and processing power of networking devices may affect the simulation results which may different from real world scenarios.

7.4 Recommendations for Future Work

The proposed system can effectively monitor the overlay network traffic in large cloud environment as the proposed system designed for single cloud to multi-cloud environment. However, there are some limitations and pending work that can be carried out as future work. The following are some suggestions that can be carried out in the future extending this work.

1. *Testing and evaluating the proposed monitoring system using real cloud systems.*

The proposed system was tested in a Linux based virtual simulated environment only. This was also highlighted as the one of the limitation of this work due to lack of resources. Therefore, it is suggested that proposed monitoring system must be tested in the real environment and their performance evaluated on a future date.

2. *Monitoring the performance of a heterogeneous cloud network environment form single cloud to multiple cloud environment.*

The simulation environment was setup with only homogeneous cloud network environment in order to create repeatable experiments. It is suggested that the environment is modified to single cloud to multiple cloud environment and the performance if the proposed system is tested as future work.

3. *Testing the system using other overlay network protocols for cloud overlay network monitoring.*

The proposed system is limited to monitor only VXLAN based overlay network technology. Since cloud overlay network technologies are still new and emerging, thus, only mature and widely adopted technology in cloud data centers Virtual eXtensible LANs (VXLAN) was considered for cloud overlay network monitoring in proposed monitoring system. This is due to the limitations of the currently available tools for cloud overlay network simulation. As future work, all other cloud overlay network technologies protocols can be added to this monitoring system.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," *Natl. Inst. Stand. Technol. Inf. Technol. Lab.*, vol. 145, p. 7, 2011.

[2] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives," *J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 2918–2933, 2014.

[3] G. Aceto, A. Botta, W. de Donato, and A. Pescapè, "Cloud monitoring: A survey," *Comput. Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[4] D. C. Marinescu, *Cloud Computing: Theory and Practice*. Elsevier Inc., 2013.

[5] S. Clayman, A. Galis, and L. Mamatas, "Monitoring virtual networks with Lattice," *Netw. Oper. Manag. Symp. Work. (NOMS Wksps), 2010 IEEE/IFIP*, no. ii, pp. 239–246, 2010.

[6] J. S. Ward and A. Barker, "Varanus: In Situ Monitoring for Large Scale Cloud Systems," *2013 IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, vol. 2, pp. 341–344, 2013.

[7] M. M. Mamta Madan, "Cloud network management model -A novel approach to manage cloud traffic," *Int. J. Cloud Comput. Serv. Archit.*, vol. 4, no. October, pp. 9–20, 2014.

[8] S. Khurram, O. Ghazali, F. Shahzad, and A. S. Osman, "A Survey of Cloud Monitoring: High Level, Low Level, Underlay and Overlay," in *NETAPPS2015*, 2015, pp. 1–7.

[9] V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle, "A Survey of Network Isolation Solutions for Multi-Tenant Data Centers," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4. pp. 2787–2821, 2016.

[10] G. T. Lakshmanan, P. Keyser, A. Slominski, F. Curbera, and R. Khalaf, "A business centric end-to-end monitoring approach for service composites," *Proc. - 2010 IEEE 7th Int. Conf. Serv. Comput. SCC 2010*, pp. 409–416, 2010.

[11] J. R. Vacca, *Computer and information security handbook*. Morgan Kaufmann Publishers, 2017.

[12] E. F. Naranjo and G. D. Salazar Ch, "Underlay and overlay networks: The approach to solve addressing and segmentation problems in the new networking era: VXLAN encapsulation with Cisco and open source networks," in *IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017*, 2018, vol. 2017-Janua, pp. 1–6.

[13] M. Mahalingam *et al.*, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," no. October, pp. 1–23, 2014.

[14] P. Garg and Y. Wang, "NVGRE: Network Virtualization Using Generic Routing Encapsulation," no. 7637, 2015.

[15] B. Davie and J. Gross, "A stateless transport tunneling protocol for network virtualization (STT)," *Draft-Davie-Stt-06*, pp. 1–21, 2012.

[16] "Cisco Global Cloud Index: Forecast and Methodology, 2013–2018," *Cisco Press*, 2014. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf.

[17] L. Santos, C. Rabadão, and R. Gonçalves, "Flow monitoring system for IoT networks," in *Advances in Intelligent Systems and Computing*, 2019, vol. 931, pp. 420–430.

[18] A. Viratanapanu, A. Kamil, A. Hamid, Y. Kawahara, and T. Asami, "On demand fine grain resource monitoring system for server consolidation," *2010 ITU-T Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services*. pp. 1–8, 2010.

[19] G. Katsaros, R. Kübert, and G. Gallizo, "Building a service-oriented monitoring framework with REST and nagios," *Proc. - 2011 IEEE Int. Conf. Serv. Comput. SCC*

2011, pp. 426–431, 2011.

[20] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "QoS-aware clouds," *Proc. - 2010 IEEE 3rd Int. Conf. Cloud Comput. CLOUD 2010*, pp. 321–328, 2010.

[21] J. Shao, H. Wei, Q. Wang, and H. Mei, "A runtime model based monitoring approach for cloud," *Proc. - 2010 IEEE 3rd Int. Conf. Cloud Comput. CLOUD 2010*, pp. 313–320, 2010.

[22] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Futur. Gener. Comput. Syst.*, vol. 28, no. 3, pp. 583–592, Mar. 2012.

[23] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 460–471, 2010.

[24] K. Alhamazani *et al.*, "CLAMS: Cross-layer Multi-cloud Application Monitoring-as-a-Service Framework," *2014 IEEE Int. Conf. Serv. Comput.*, no. iii, pp. 283–290, 2014.

[25] Y.-S. Peng and Y.-C. Chen, "SNMP-based monitoring of heterogeneous virtual infrastructure in clouds," *13th Asia-Pacific Netw. Oper. Manag. Symp.*, pp. 1–6, 2011.

[26] M. Roughan, "A Case Study of the Accuracy of SNMP Measurements," *J. Electr. Comput. Eng.*, vol. 2010, pp. 1–7, Jan. 2010.

[27] P. Barford and J. Sommers, "Comparing probe-and router-based packet-loss measurement," *IEEE Internet Comput.*, vol. 8, no. 5, pp. 50–56, Sep. 2004.

[28] L. Deri and F. Fusco, "MicroCloud-based network traffic monitoring," p. 1418.

[29] V. Mann, A. Vishnoi, and S. Bidkar, "Living on the edge: Monitoring network flows at the edge in cloud data centers," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, 2013, pp. 1–9.

[30] "NetFlow." [Online]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html. [Accessed: 25-Oct-2017].

[31] P. Phaal, S. Panchen, and N. McKee, “InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” *Rfc 3176*, pp. 1–31, 2001.

[32] P. Aitken, “Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information,” pp. 1–75, Sep. 2013.

[33] P. Kopietz, P. Scharf, M. Skaf, and S. Chakravarty, “Requirements for IP Flow Information Export (IPFIX),” *IETF RFC 3917*, pp. 1–34, 2004.

[34] T. Zseby, E. Boschi, N. Brownlee, and B. Claise, *IP Flow Information Export (IPFIX) Applicability*. 2009, pp. 1–31.

[35] N. Bitar, S. Gringeri, and J. X. Tiejun, “Technologies and protocols for data center and cloud networking,” *IEEE Commun. Mag.*, vol. 51, no. September, pp. 24–31, Sep. 2013.

[36] A. A. Semnanian, J. Pham, B. Englert, and X. Wu, “Virtualization technology and its impact on computer hardware architecture,” in *Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011*, 2010, pp. 719–724.

[37] F. Bazargan, C. Y. Yeun, and M. J. Zemerly, “State-of-the-Art of Virtualization , its Security Threats and Deployment Models,” vol. 2, no. December, pp. 335–343, 2012.

[38] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[39] “VXLAN Overview: Cisco Nexus 9000 Series Switches - Cisco.” [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-729383.html. [Accessed: 12-Jul-2015].

[40] “NVGRE – Define The Cloud.” [Online]. Available: http://www.definethecloud.net/nvgre/. [Accessed: 06-Feb-2015].

[41] “Comparison: VXLAN vs NVGRE vs STT vs LISP - Overlay Network Technologies.” [Online]. Available: https://www.routexp.com/2017/06/comparison-

vxlan-vs-nvgre-vs-stt-vs.html. [Accessed: 30-Aug-2017].

[42] Y. Mei, L. Liu, X. Pu, and S. Sivathanu, "Performance measurements and analysis of network I/O applications in virtualized cloud," in *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, 2010, pp. 59–66.

[43] "OpenNebula | Flexible Enterprise Cloud Made Simple." [Online]. Available: http://opennebula.org/. [Accessed: 08-Nov-2017].

[44] "Nagios Core. Nagios Open Source Project." [Online]. Available: https://www.nagios.org/. [Accessed: 08-Nov-2017].

[45] "Nimbus." [Online]. Available: http://www.nimbusproject.org/. [Accessed: 10-Nov-2017].

[46] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, "GMonE: A complete approach to cloud monitoring," *Futur. Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2026–2040, 2013.

[47] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds," *Futur. Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2041–2056, 2013.

[48] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 130–137, 2011.

[49] M. Rak, S. Venticinque, T. Máhr, G. Echevarria, and G. Esnal, "Cloud application monitoring: The mOSAIC approach," *Proc. - 2011 3rd IEEE Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2011*, pp. 758–763, 2011.

[50] V. C. Emeakaroha, T. C. Ferreto, M. a S. Netto, I. Brandic, and C. a F. De Rose, "CASViD: Application level monitoring for SLA violation detection in clouds," *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 499–508, 2012.

[51] "Amazon CloudWatch - Cloud & Network Monitoring Services." [Online]. Available:

https://aws.amazon.com/cloudwatch/.

[52] "Monitoring and Autoscaling features for Windows Azure with AzureWatch indepth - AzureWatch." [Online]. Available: https://www.paraleap.com/AzureWatch. [Accessed: 01-Jan-2016].

[53] "Nimsoft." [Online]. Available: http://www.nimsoft.com. [Accessed: 08-Nov-2017].

[54] H. Patel and A. Meniya, "A survey on commercial and open source cloud monitoring," *Int. J. Sci. Mod.*, vol. 1, no. 2, pp. 42–44, 2013.

[55] S. Bardhan and D. Milojicic, "A mechanism to measure quality-of-service in a federated cloud environment," *Proc. 2012 Work. Cloud Serv. Fed. 8th open cirrus summit - Fed. '12*, pp. 19–24, 2012.

[56] Y. Lee, "QoS metrics for service level measurement for SOA environment," *2010 6th Int. Conf. Adv. Inf. Manag. Serv.*, pp. 509–514, 2010.

[57] C.-H. L. Duo Liu, Utkarsh Kanabar, "A light weight SLA management infrastructure for cloud computing," *Can. Ieee Of, Conf. Eng. Electr.*, pp. 1–4, 2013.

[58] W. Stallings, *SNMP,SNMPV2,Snmpv3,and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[59] R. Grati and K. Boukadi, "A framework for IaaS-to-SaaS monitoring of BPEL processes in the Cloud : design and evaluation," pp. 557–564, 2014.

[60] K. Alhamazani, R. Ranjan, F. Rabhi, L. Wang, and K. Mitra, "Cloud monitoring for optimizing the QoS of hosted applications," *CloudCom 2012 - Proc. 2012 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, vol. 1, pp. 765–770, 2012.

[61] L. Ye, H. Zhang, J. Shi, and X. Du, "Verifying Cloud Service Level Agreement," *2012 IEEE Glob. Commun. Conf.*, pp. 777–782, 2012.

[62] O. Adinolfi, R. Cristaldi, L. Coppolino, and L. Romano, "QoS-MONaaS: A portable architecture for QoS monitoring in the cloud," *8th Int. Conf. Signal Image Technol. Internet Based Syst. SITIS 2012r*, pp. 527–532, 2012.

[63] M. Dhingra, J. Lakshmi, and S. K. Nandy, “Resource usage monitoring in clouds,” *Proc. - IEEE/ACM Int. Work. Grid Comput.*, vol. 012, pp. 184–191, 2012.

[64] L. Kai, T. Weiqin, Z. Liping, and H. Chao, “SCM: A design and implementation of monitoring system for CloudStack,” *Proc. - 2013 Int. Conf. Cloud Serv. Comput. CSC 2013*, pp. 146–151, 2013.

[65] C. Rodrigues, L. Z. Granville, and L. M. R. Tarouco, “Network and Services Monitoring : A Survey in Cloud Computing Environments,” no. c, pp. 7–13, 2012.

[66] E. Systems, “Eucalyptus - Cloud Computing Platform User Guide.” [Online]. Available: http://www.eucalyptus.com.

[67] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, and T. Varvarigou, “A Self-adaptive hierarchical monitoring mechanism for Clouds,” *J. Syst. Softw.*, vol. 85, no. 5, pp. 1029–1041, 2012.

[68] Y. He, X. Wang, Y. Chen, Z. Du, W. Huang, and X. Chai, “A simulation cloud monitoring framework and its evaluation model,” *Simul. Model. Pract. Theory*, vol. 38, pp. 20–37, 2013.

[69] a. Meera and S. Swamynathan, “Agent based Resource Monitoring System in IaaS Cloud Environment,” *Procedia Technol.*, vol. 10, pp. 200–207, 2013.

[70] Libvirt, “libvirt - The virtualization API,” 2014. [Online]. Available: http://libvirt.org/index.html. [Accessed: 24-Oct-2017].

[71] M. Smit, B. Simmons, and M. Litoiu, “Distributed, application-level monitoring for heterogeneous clouds using stream processing,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2103–2114, 2013.

[72] G. Suciu, S. Halunga, A. Ochian, and V. Suciu, “Network Management and Monitoring for Cloud Systems,” pp. 1–4, 2014.

[73] M. Andreolini, M. Colajanni, and M. Pietri, “A Scalable Architecture for Real-Time Monitoring of Large Information Systems,” *2012 Second Symp. Netw. Cloud Comput.*

Appl., pp. 143–150, 2012.

[74] Y. M. Kim *et al.*, "Architecture of a network performance monitor for application services on multi-clouds," *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 594–599, 2013.

[75] M. K. Nair and V. Gopalakrishna, "'CloudCop': Putting network-admin on cloud nine: Towards cloud computing for network monitoring," in *IEEE International Conference on Internet Multimedia Services Architecture and Applications, IMSAA*, 2009, pp. 1–6.

[76] G. Liu and T. Wood, "Cloud-Scale Application Performance Monitoring with SDN and NFV," *2015 IEEE Int. Conf. Cloud Eng.*, pp. 440–445, 2015.

[77] V. Mann, A. Vishnoi, and S. Bidkar, "Living on the edge: Monitoring network flows at the edge in cloud data centers," *2013 Fifth Int. Conf. Commun. Syst. Networks*, pp. 1–9, 2013.

[78] "NetFlow." [Online]. Available: http://www.cisco.com/go/netflow/. [Accessed: 22-Oct-2017].

[79] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," *Rfc 3176*, no. 3176, pp. 1–31, Sep. 2001.

[80] L. Deri, "MicroCloud-based Network Traffic Monitoring," pp. 864–867, 2013.

[81] M. Scharf *et al.*, "Monitoring and abstraction for networked clouds," *2012Proceedings Intell. Next Gener. Networks, ICIN 2012*, pp. 80–85, 2012.

[82] R. Alimi, Y. Yang, and R. Penno, "Application-Layer Traffic Optimization (ALTO) Deployment," *RFC7285*, 2014. [Online]. Available: https://tools.ietf.org/html/rfc7285.

[83] T. Mullins and A. Bagula, "Monitoring community clouds: The lightweight network management protocol," *Proc. - IEEE 10th Int. Conf. Ubiquitous Intell. Comput. UIC*

2013 IEEE 10th Int. Conf. Auton. Trust. Comput. ATC 2013, pp. 678–684, 2013.

[84] "SIGAR." [Online]. Available: https://support.hyperic.com/display/SIGAR/Home.

[85] J. Moses, R. Iyer, R. Illikkal, S. Srinivasan, and K. Aisopos, "Shared resource monitoring and throughput optimization in cloud-computing datacenters," *Proc. - 25th IEEE Int. Parallel Distrib. Process. Symp. IPDPS 2011*, pp. 1024–1033, 2011.

[86] K. Ma, R. Sun, and A. Abraham, "Toward a lightweight framework for monitoring public clouds," *Proc. 2012 4th Int. Conf. Comput. Asp. Soc. Networks, CASoN 2012*, pp. 361–365, 2012.

[87] J. Shao and Q. Wang, "A performance guarantee approach for cloud applications based on monitoring," *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 25–30, 2011.

[88] C. Robitaille, "Network Performance: Active Monitoring? Passive Monitoring? How About Both?" [Online]. Available: https://accedian.com/service-providers/blog-2/network-performance-active-passive-monitoring/. [Accessed: 04-Jul-2018].

[89] M. S. Aktas, "Hybrid cloud computing monitoring software architecture," *Concurr. Comput. Pract. Exp.*, vol. 30, no. 21, p. e4694, Nov. 2018.

[90] C. Hare, "Simple Network Management Protocol (SNMP)," *Encycl. Inf. Assur.*, pp. 2721–2727, 2016.

[91] "SNMP (Simple Network Management Protocol)." [Online]. Available: https://support.nagios.com/kb/article/snmp-simple-network-management-protocol-49.html. [Accessed: 15-Mar-2015].

[92] L. Andrey, O. Festor, A. Lahmadi, A. Pras, and J. Schönwälder, "Survey of SNMP performance analysis studies," *Int. J. Netw. Mgmt*, vol. 19, no. April 2009, pp. 527–548, Nov. 2009.

[93] D. Fernandez, H. Lorenzo, F. J. Novoa, F. Cacheda, and V. Carneiro, "Tools for managing network traffic flows: A comparative analysis," in *IEEE 16th International Symposium on Network Computing and Applications, NCA 2017*, 2017, vol. 2017-

Janua, pp. 1–5.

[94] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 339–350, Aug. 2006.

[95] "SFlow vs NetFlow vs SNMP: What Are the Differences?" [Online]. Available: http://www.cables-solutions.com/sflow-vs-netflow-vs-snmp-differences.html. [Accessed: 01-Jul-2018].

[96] A. C. Myers, "JFlow," in *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '99*, 1999, pp. 228–241.

[97] R. Grezo and M. Nagy, "Network traffic measurement and management in software defined networks," in *3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 541–546.

[98] "NetFlow vs. sFlow vs. IPFIX vs. NetStream. Network Traffic Analysis and Network Traffic Monitoring explained." [Online]. Available: https://www.noction.com/blog/netflow-sflow-ipfix-netstream-network-traffic-monitoring-explained. [Accessed: 01-Jul-2018].

[99] A. Kobayashi and B. Claise, "IP Flow Information Export (IPFIX) Mediation: Problem Statement," Aug. 2010.

[100] L. T. M. Blessing and A. Chakrabarti, "DRM, a Design Research Methodology," *DRM, a Des. Res. Methodol.*, pp. 1–12, 2009.

[101] H. Birkhofer, "Are We Aware of What We Are Doing in Design Research?," *3rd Int. Conf. Des. Eng. Sci. ICDES 2014, Pilsen, Czech Repub.*, pp. 1–10, 2014.

[102] V. Jacobson and S. McCanne, "libpcap: Packet capture library," *Lawrence Berkeley Lab. Berkeley, CA*, 2009.

[103] "YAF - Yet Another Flowmeter." [Online]. Available: https://tools.netsa.cert.org/yaf/. [Accessed: 25-Sep-2017].

[104] P. Velan, "Practical Experience with IPFIX Flow Collectors," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 1021–1026.

[105] "SiLK," *(CERT NetSA) Carnegie Mellon University*. [Online]. Available: https://tools.netsa.cert.org/silk/. [Accessed: 25-Oct-2017].

[106] "Mininet: An Instant Virtual Network on your Laptop or PC - Mininet." [Online]. Available: http://mininet.org/. [Accessed: 23-Sep-2017].

[107] "Open vSwitch." [Online]. Available: http://openvswitch.org/. [Accessed: 23-Sep-2017].

[108] P. Morreale and J. Anderson, "OpenFlow," *Softw. Defin. Netw.*, vol. 38, no. 2, pp. 107–128, Mar. 2014.

[109] D. K. Pace, "Verification, Validation, and Accreditation of Simulation Models," in *Applied System Simulation*, 2011, pp. 487–506.

[110] T. L. Paez, "Introduction to model validation," in *27th International Modal Analysis Conference*, 2009, pp. 1–11.

[111] L. Versluis, M. Neacsu, and A. Iosup, "A Trace-Based Performance Study of Autoscaling Workloads of Workflows in Datacenters," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2018, pp. 223–232.

[112] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." [Online]. Available: https://iperf.fr/. [Accessed: 25-Sep-2017].

[113] J.-M. Kelif, S. Senecal, M. Coupechoux, and C. Bridon, "Analytical performance model for Poisson wireless networks with pathloss and shadowing propagation," in *2014 IEEE Globecom Workshops (GC Wkshps)*, 2014, pp. 1528–1532.

[114] I. A. Lawal, A. Md Said, and A. A. Mu'azu, "Simulation model to improve QoS performance over fixed WiMAX using OPNET," *Res. J. Appl. Sci. Eng. Technol.*, vol.

6, no. 21, pp. 3933–3945, 2013.

[115] M. Ivanovich *et al.*, "Measuring Quality of Service in an Experimental Wireless Data Network," *Aust. Telecommun. Networks Appl. Conf.*, vol. perth, Australia, 2003.

[116] M. S. (Mohammad S. Obaidat, N. Boudriga, and Wiley InterScience (Online service), *Fundamentals of performance evaluation of computer and telecommunications systems*. John Wiley & Sons, 2010.

[117] K. Velten, *Mathematical modeling and simulation: introduction for scientists and engineers*. 2009.

[118] K. Soetaert, T. Petzoldt, and R. W. Setzer, "Solving differential equations in R," *R J.*, 2010.

[119] J. Anzures-Cabrera and J. P. T. Higgins, "Graphical displays for meta-analysis: An overview with suggestions for practice," *Res. Synth. Methods*, vol. 1, no. 1, pp. 66–80, 2010.

[120] R. Jain, *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. 1991.

[121] K. Goga, O. Terzo, P. Ruiu, and F. Xhafa, "Simulation, Modeling, and Performance Evaluation Tools for Cloud Applications," in *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, 2014, pp. 226–232.

[122] M. S. Obaidat, F. Zarai, and P. Nicopolitidis, *Modeling and simulation of computer networks and systems : methodologies and applications*. sciencedirect, 2015.

[123] M. N. ISMAIL, "Comparing the Accuracy of Network Utilization Performance between Real Network and Simulation Model for Local Area Network (LAN)," *Int'l J. Commun. Netw. Syst. Sci.*, vol. 01, no. 04, pp. 339–349, 2009.

[124] R. Sherwood *et al.*, "Can the production network be the testbed?," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation, OSDI, USENIX Association*, 2010, vol. 10, pp. 1–6.

[125] S. B. Lee *et al.*, "Improving Simulation for Network Research," *Univ. South. Calif.*, 1999.

[126] D. Muelas, J. Ramos, and J. E. L. de Vergara, "Assessing the Limits of Mininet-Based Environments for Network Experimentation," *IEEE Netw.*, vol. 32, no. 6, pp. 168–176, Nov. 2018.

[127] F. Keti and S. Askar, "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments," in *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, 2015, vol. 2015-Octob, pp. 205–210.

[128] J. Janitor, F. Jakab, and K. Kniewald, "Visual Learning Tools for Teaching/Learning Computer Networks: Cisco Networking Academy and Packet Tracer," in *2010 Sixth International Conference on Networking and Services*, 2010, pp. 351–355.

[129] C. Welsh, *GNS3 network simulation guide*. Packet Publishing, 2013.

[130] Shie-Yuan Wang, Chih-Liang Chou, and Chun-Ming Yang, "EstiNet openflow network simulator and emulator," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 110–117, Sep. 2013.

[131] J. Suarez-Varela and P. Barlet-Ros, "Towards a NetFlow Implementation for OpenFlow Software-Defined Networks," in *Proceedings of the 29th International Teletraffic Congress, ITC 2017*, 2017, vol. 1, pp. 187–195.

[132] M. Hassan and R. Jain, *High performance TCP/IP networking : concepts, issues, and solutions*. Pearson/Prentice Hall, 2004.

[133] S. Frey, L. Claudia, and C. Reich, "Key Performance Indicators for Cloud Computing SLAs," in *International Conference on Emerging Network Intelligence*, 2013, pp. 60–64.

[134] A. Hanemann, A. Liakopoulos, M. Molina, and D. M. Swany, "A study on network performance metrics and their composition," *Campus-Wide Inf. Syst.*, vol. 23, no. 4, pp. 268–282, 2006.

[135] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Futur. Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023, 2013.

[136] IETF, "IP Flow Information Export (ipfix)." [Online]. Available: http://datatracker.ietf.org/wg/ipfix/charter/. [Accessed: 23-May-2017].

[137] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices," *RFC 2544*, 1999.

[138] J. Weber, "The Fundamentals of Passiv e Monitoring Access Net Optics Learning Center Presents," 2006. [Online]. Available: https://www.nanog.org/meetings/nanog37/presentations/joy-weber.pdf. [Accessed: 25-Oct-2017].

[139] Z. Jian and A. Moore, "Traffic trace artifacts due to monitoring via port mirroring," in *2007 Workshop on End-to-End Monitoring Techniques and Services*, 2007, pp. 1–8.

[140] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending Networking into the Virtualization Layer.," in *8th ACM Workshop on Hot Topics inNetworks*, 2009, vol. VIII, pp. 1–6.

[141] L. Braun, A. Didebulidze, N. Kammenhuber, and G. Carle, "Comparing and improving current packet capturing solutions based on commodity hardware," in *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, 2010, pp. 206–217.

[142] L. Deri, "PF_Ring," 2012. [Online]. Available: http://www.ntop.org/products/packet-capture/pf_ring/. [Accessed: 09-Sep-2017].

[143] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection Status," 2009.

[144] M. Mahalingam *et al.*, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," Aug. 2014.

[145] "VXLAN Packet header detail." [Online]. Available: http://soa.sys-con.com/node/2929732/mobile. [Accessed: 06-Dec-2014].

[146] "VXLAN MTU vs IP MTU Consideration - ipengineer.net." [Online]. Available: http://ipengineer.net/2014/06/vxlan-mtu-vs-ip-mtu-consideration/. [Accessed: 10-Jan-2015].

[147] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 280–292, Jun. 2001.

[148] B. Trammell and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)," 2018.

[149] G.Sadasivan, N.Brownlee, B.Claise, and J.Quittek, "Architecture for IP Flow Information Export," RFC Editor, 2009.

[150] A. Neumann, M. Ehrlich, L. Wisniewski, and J. Jasperneite, "Towards monitoring of hybrid industrial networks," in *IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, 2017, pp. 1–4.